Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2009

Miguel da Rosa
Carvalhal Sequeira

Percepção e localização inteligente para condução autónoma

Perception and intelligent localization for autonomous driving

"The truth is rarely pure and never simple."

— Oscar Wilde

Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2009

Miguel da Rosa
Carvalhal Sequeira

Percepção e localização inteligente para condução autónoma

Perception and intelligent localization for autonomous driving

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado Integrado em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Dr. Artur José Carneiro Pereira e do Dr. José Luís Costa Pinto de Azevedo, Professores Auxiliares do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri / the jury

presidente / president

Doutor Tomás António Mendes Oliveira e Silva
Professor Associado da Universidade de Aveiro

vogais / examiners committee

Doutor Agostinho Gil Teixeira Lopes
Investigador Auxiliar da Universidade do Minho

Doutor Artur José Carneiro Pereira
Professor Auxiliar da Universidade de Aveiro (orientador)

Doutor José Luís Costa Pinto de Azevedo
Professor Auxiliar da Universidade de Aveiro (co-orientador)

agradecimentos / acknowledgements

Queria aproveitar para agradecer essencialmente aos meus pais e à minha irmã, por tudo. Também quero agradecer ao meu orientador e co-orientador, assim como aos amigos mais próximos.

Resumo

Visão por computador e fusão sensorial são temas relativamente recentes, no entanto largamente adoptados no desenvolvimento de robôs autónomos que exigem adaptabilidade ao seu ambiente envolvente. Esta dissertação foca-se numa abordagem a estes dois temas para alcançar percepção no contexto de condução autónoma. O uso de câmaras para atingir este fim é um processo bastante complexo. Ao contrário dos meios sensoriais clássicos que fornecem sempre o mesmo tipo de informação precisa e atingida de forma determinística, as sucessivas imagens adquiridas por uma câmara estão repletas da mais variada informação e toda esta ambígua e extremamente difícil de extrair. A utilização de câmaras como meio sensorial em robótica é o mais próximo que chegamos na semelhança com aquele que é o de maior importância no processo de percepção humana, o sistema de visão. Visão por computador é uma disciplina científica que engloba áreas como: processamento de sinal, inteligência artificial, matemática, teoria de controlo, neurobiologia e física.

A plataforma de suporte ao estudo desenvolvido no âmbito desta dissertação é o ROTA (RObô Triciclo Autónomo) e todos os elementos que consistem o seu ambiente. No contexto deste, são descritas abordagens que foram introduzidas com fim de desenvolver soluções para todos os desafios que o robô enfrenta no seu ambiente: detecção de linhas de estrada e consequente percepção desta, detecção de obstáculos, semáforos, zona da passadeira e zona de obras. É também descrito um sistema de calibração e aplicação da remoção da perspectiva da imagem, desenvolvido de modo a mapear os elementos percepcionados em distâncias reais. Em consequência do sistema de percepção, é ainda abordado o desenvolvimento de auto-localização integrado numa arquitectura distribuída incluindo navegação com planeamento inteligente. Todo o trabalho desenvolvido no decurso da dissertação é essencialmente centrado no desenvolvimento de percepção robótica no contexto de condução autónoma.

Abstract

Computer vision and sensor fusion are subjects that are quite recent, however widely adopted in the development of autonomous robots that require adaptability to their surrounding environment. This thesis gives an approach on both in order to achieve perception in the scope of autonomous driving. The use of cameras to achieve this goal is a rather complex subject. Unlike the classic sensorial devices that provide the same type of information with precision and achieve this in a deterministic way, the successive images acquired by a camera are replete with the most varied information, that this ambiguous and extremely difficult to extract. The use of cameras for robotic sensing is the closest we got within the similarities with what is of most importance in the process of human perception, the vision system. Computer vision is a scientific discipline that encompasses areas such as signal processing, artificial intelligence, mathematics, control theory, neurobiology and physics.

The support platform in which the study within this thesis was developed, includes ROTA (RObô Triciclo Autónomo) and all elements comprising its environment. In its context, are described approaches that introduced in the platform in order to develop solutions for all the challenges facing the robot in its environment: detection of lane markings and its consequent perception, obstacle detection, traffic lights, crosswalk and road maintenance area. It is also described a calibration system and implementation for the removal of the image perspective, developed in order to map the elements perceived in actual real world distances. As a result of the perception system development, it is also addressed self-localization integrated in a distributed architecture that allows navigation with long term planning. All the work developed in the course of this work is essentially focused on robotic perception in the context of autonomous driving.

Contents

Contents i

List of Figures iii

1 Introduction 1
- 1.1 Autonomous Driving . . . 1
- 1.2 History and State of the Art . . . 4
- 1.3 Experimental Platform . . . 11
- 1.4 Previous Work . . . 16
- 1.5 Motivation . . . 18
- 1.6 Goals . . . 19
- 1.7 Achievements . . . 21
- 1.8 Tools Used . . . 22
- 1.9 Thesis Structure . . . 24

2 Vision Feature Extraction - Low Level Processing 25
- 2.1 Edge Segmentation . . . 26
- 2.2 Color Segmentation . . . 28
- 2.3 Template Matching . . . 30
- 2.4 Image Morphology . . . 33

3 Sensing Information, Fusion and Estimation 37
- 3.1 Linear Regression . . . 38
- 3.2 Non-Linear Regression . . . 41
- 3.3 Kalman Filter . . . 43
- 3.4 Extended - Kalman Filter . . . 48

4 Visual Perception System 51
- 4.1 Lane Detection System . . . 52
- 4.2 Traffic Lights Detection . . . 64
- 4.3 Road Maintenance Area Detection . . . 65
- 4.4 Obstacle Detection . . . 68
- 4.5 Inverse Perspective Mapping . . . 69

5 Self Localization 75
- 5.1 System Architecture . . . 77
- 5.2 Car Pose in a Local Road Section . . . 82

5.3 Car Dynamics Model 84
5.4 Car Pose in a Relative Coordinate System 88

6 Conclusion and Future Work **95**

Bibliography **97**

List of Figures

1.1 Basic Autonomous React Agent . . . 2
1.2 Grid of pixels in an image. . . . 3
1.3 VAMP . . . 4
1.4 Lane marker extraction using dark-light-dark lane profiles. . . . 5
1.5 BRAiVE . . . 6
1.6 The Gold System IPM. . . . 7
1.7 Navlab 11 . . . 8
1.8 RALPH Screenshot . . . 9
1.9 RALPH Overview . . . 10
1.10 Environment . . . 12
1.11 Traffic Signs . . . 13
1.12 ROTA . . . 14
1.13 ROTA: low-level hardware architecture. . . . 14
1.14 ROTA: FireWire cameras. . . . 15
1.15 ROTA: rear camera. . . . 15
1.16 ROTA: front camera. . . . 16
1.17 ROTA: Behaviors. . . . 17
1.18 ROTA: Image Sensors. . . . 18
1.19 Perceptual Reasoning System Proposed Architecture . . . 21
1.20 Git Version Control. . . . 23
1.21 Doxygen Documentation. . . . 23

2.1 ROI - Region of Interest. . . . 26
2.2 Canny algorithm demonstration . . . 27
2.3 HSV Cone . . . 30
2.4 Color Segmentation example . . . 30
2.5 Template Matching example . . . 31
2.6 Template Variances . . . 33
2.7 Morphology operation example . . . 35

3.1 Vertical deviation illustration . . . 39
3.2 Kalman Cycle . . . 45
3.3 Kalman Complete Cycle . . . 47

4.1 Lane Detection, Region of Interest example. . . . 52
4.2 Global Lane Markings Search. . . . 54
4.3 Lane Detection Examples . . . 57

4.4 Estimation for the curvature analysis. 61
4.5 Error estimation and analysis. 62
4.6 Estimation analysis for each lane marking. 63
4.7 Template Signs example. 64
4.8 Traffic Lights Detection example. 65
4.9 Traffic Lights Detection Competition example. 66
4.10 Cones Segmentation example. 67
4.11 Road Maintenance Detection example. 67
4.12 Obstacle Detection example. 68
4.13 Bi-linear Interpolation example. 71
4.14 Inverse Perspective Maps. 72
4.15 IPM Calibration Method. 72
4.16 Lane Detection and Perspective Removal. 73

5.1 The Proposed System Conceptual Model 77
5.2 The Proposed System Class Model 79
5.3 The track plant 80
5.4 The Track Nodes 81
5.5 Example for Way-points Generation. 81
5.6 Local Car Poses Coordinate System 82
5.7 Local Car Pose in a Straight Section 83
5.8 Local Car Pose in a Curved Section. 85
5.9 Robot Curvature Coordinate System. 85
5.10 Car Orientation 86
5.11 Car Dynamics in a Rectilinear Movement. 87
5.12 Car Dynamics in a Curvilinear Movement. 87
5.13 Observation Model in a Straight Section. 92
5.14 Observation Model in a Curved Section. 92

Chapter 1

Introduction

1.1 Autonomous Driving

In the 1989 film *Batman*, the Bat-mobile is shown to be able to drive itself to Batman's current location somewhere in Gotham City. In order to do this, the car had to go from one point to another without any assistance from a driver. Also, being in a urban environment, the Bat-mobile would need to:

1. Understand its immediate environment (Sensing and Perception)
2. Know where it is and where it wants to go (Localization and Navigation)
3. Find its way in the traffic (Motion planning)
4. Operate the mechanics of the vehicle (Actuation)

Systems that aim at possessing these four capabilities are often called Autonomous Driving Systems. The development of this type of systems is a challenging topic that gained the interest of research institutions all across the world since mid eighties. One motivation is that each year there are thousands of car accidents in the whole world, accidents that claim many lives and cost billions. A robust intelligent driver system would be collision and accident free. There are more advantages to autonomous driving besides the increased safety and accident rate reduction. Decreasing fuel expenses and consequently pollution by automatically controlling the vehicles velocity keeping a soft acceleration profile or usage of intelligent car routing in order to decrease the number of traffic jams/congestions are other advantages. Intelligent vehicles could also prove useful in our aging society. Older people are known to have some difficulties while driving and an intelligent system could help them as well as people with disabilities. Furthermore, the military has shown increasing interest in intelligent vehicles, scout vehicles typically operate behind enemy lines, becoming an easy target. The desire to have automated scouts that can investigate hazardous areas without putting soldiers in jeopardy has long been shared by military forces. Currently, the demand for intelligent vehicles is increasing, and the technologies to build them are becoming more available. Advances on this field have been driven by the military, aerospace, and electronics industry. Even the personal entertainment industry has helped, with the proliferation of camcorders driving down the cost

of video sensors. Autonomous mobile systems are expected to become, in the near future, a common presence in dynamic, structured or unstructured environments.

To be autonomous, a mobile system must be able to react to environment changes. In order to do that, a system must use sensors to create a perception of the state of the world, and then use actuators to perform some action that will affect, in some way, its posterior state on the environment, or the state of the environment itself. This is the basis of an autonomous agent as illustrated in Fig. 1.1. Sensors can be lasers, video cameras, GPS, etc. Laser sensors are very accurate and allow to obtain a precisely measured 3D representation of world (coordinates in real distances). However, lasers are extremely expensive sensors, while video cameras are not, and this is the main reason why they are not used in this project. Based on video cameras we can have monocular vision, by using only one camera, a 360° view with omni-cameras or stereoscopic vision (with two or more cameras). Using two cameras and stereoscopic algorithms, one can achieve distance perception. Still, despite the lack of accurate measuring, video sensors are still a premium choice because of their accessible price and versatility. The science of manipulating images using video cameras as sensors to build "machines that see" is quite recent, with no more than 30 years. This recent field of sciences, Computer Vision, comprises a major part of the presented thesis.

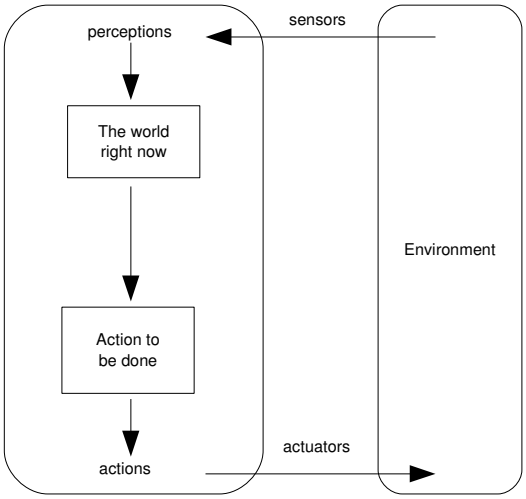


Figure 1.1: Basic Autonomous React Agent.

Computer Vision is a vast field and the most used technology in the autonomous driving field. It addresses tasks like determining whether or not an image data contains some specific object, feature, or activity. This task that can normally be solved robustly and without effort by a human, is still not satisfactorily solved in computer vision for the general case (arbitrary objects in arbitrary situations). The existing methods for dealing with this problem can, at best, solve it only for objects in specific situations: human faces, printed or hand-written characters, vehicles. These situations are typically described in terms of well-defined illumination, background and pose of the object relative to the camera. This task becomes more complex when object's properties, like shape, size or color, change with pose and lighting conditions. These factors changes, represent a significant problem particularly on outdoor

scenarios, due to their unpredictable nature. Variations in the world (weather, lighting, reflections, movements), imperfections in the lens and mechanical setup, electrical noise in the sensors or other electronics and compression artifacts after image capture, all add up to the already difficult task of determining if the image data contains any specific desired feature.

This immature science, Computer Vision, is a complex and difficult subject, although, at a first glance that can be misleading. This, is due to the ability that we, humans, have to perform this kind of task on a everyday basis. The fact is that the human brain divides the vision signal into many channels that stream different kinds of information. The brain has an attention system that identifies, in a task-dependent way, important parts of an image to examine while suppressing examination of other areas. There is massive feedback in the visual stream that is, as of yet, little understood. There are widespread associative inputs from muscle control sensors and all of the other senses that allow the brain to draw on cross-associations made from years of living in the world. The feedback loops in the brain go back to all stages of processing including the hardware sensors themselves (the eyes), which mechanically control lighting via the iris and tune the reception on the surface of the retina.

In a machine vision system, however, a computer receives a grid of numbers from the camera or from a disk, and that is it. There is no built-in pattern recognition, no automatic control of focus and aperture, and no cross-associations with years of experience. While in a picture we may see a car, what the computer sees is just a grid of numbers. Any given number within that grid has a rather large noise component and by itself does not provide a large amount of information, however, that is all that the computer sees, see Fig.1.2. Our task then becomes to turn this noisy grid of numbers into an visual perception. Due to complexity and massive data rate, usually constrained under real-time requirements, computer vision systems are usually multi-staged. Stages may include: image acquisition, pre-processing, feature extraction, element detection and high level processing.

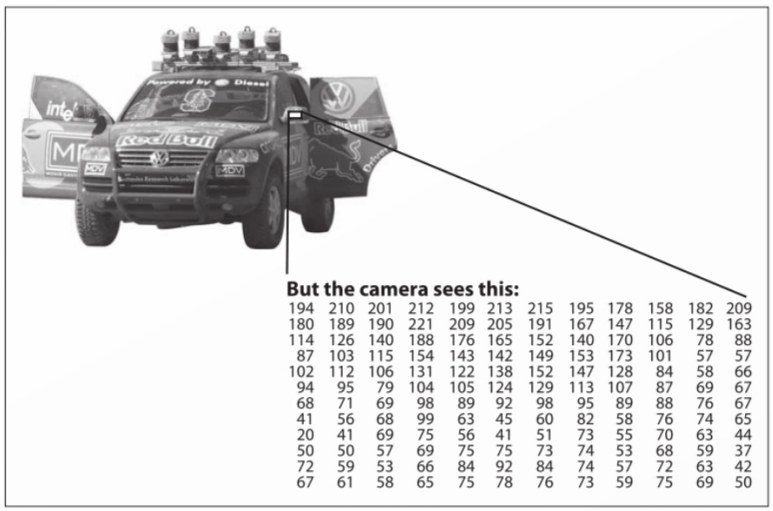


Figure 1.2: To a computer, the car's side mirror is just a grid of number, [11].

1.2 History and State of the Art

The initial phase of this thesis work required research on the autonomous driving state of the art, the result being a snapshot of the autonomous driving history and achievements, analysing cases of success and technologies that could prove useful in our context. Previously, in this thesis, the concept of Autonomous Driving Systems was introduced, followed by the science of Computer Vision. Now will be addressed the history and state of the art of autonomous driving, referring the work done over the years by some research groups that developed, and still develop autonomous vehicles. For each one of this groups, their achievements will be described, followed by their major contributions to the area.

There are some steps towards autonomous driving given in the past that are worth mentioning. One important man who gave some of these steps was Ernst Dickmanns, a pioneer of dynamic computer vision and driverless cars (readings of his work can be found on his book [23]). In 1980, he gathered his team at the Universität der Bundeswehr in Munich (Germany) and achieved autonomous driving with a modified vision-guided Mercedes-Benz robot van ("VaMoRs" was the name of the project). This great achievement gathered interest from the European Commission that began funding Dickmanns' team under the 800 million Euro EUREKA Promotheus Project on autonomous vehicles ("PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety, 1987-1995). In 1994-1995, Ernst Dickmanns engineered VaMP and Vita-2 of Daimler-Benz and re-engineered S-Class Mercedes-Benz VaMoRs. The first two cars drove more than one thousand kilometers on a Paris three-lane highway in standard heavy traffic at speeds up to 130 Km/h; the Mercedes-Benz took a 1600 Km trip from Munich to Copenhagen and back, achieving speeds exceeding 175 Km/h. This cars demonstrated autonomous driving in free lanes, convoy driving, and lane changes left and right with autonomous passing of other cars.

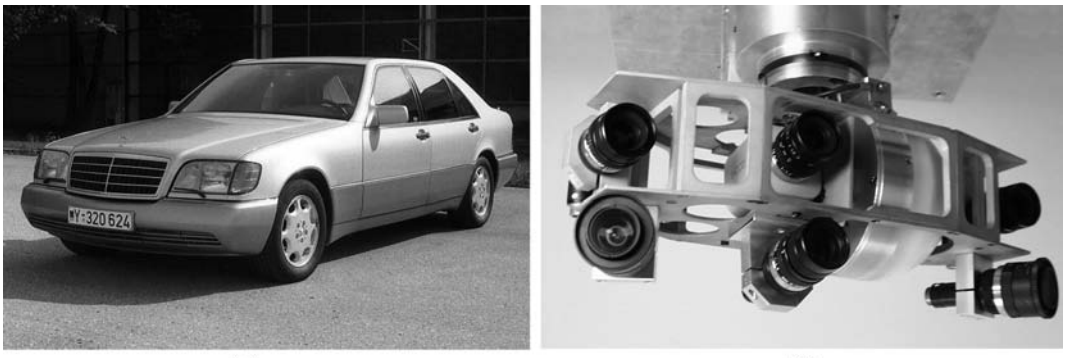


Figure 1.3: VAMP, one of the vehicles developed by Dickmanns's team. All the three cars (VAMP,VITA-2,VAMOR) were equipped with one version of the MarVEye, a very interesting approach in vehicle sensing equipment. (a) VAMP vehicle overall look (b) MarVEye system, [23].

Despite of Dickmann's advances, all his cars still were semi-autonomous and a part of the trip required human interventions. Nevertheless Dickmanns's team made great contributions to the area, one of them being what they called the 4-D approach [22]. In their approach,

they used spatiotemporal models for the fourth dimension supported by Kalman filters, by incorporating a model of the world in space (3D) and in time (1D) that is recursively refined. This made possible to yield estimates of the vehicle and road state parameters in a predefined look ahead range.

The vision system used by Dickmanns's vehicles, [20], performs as follows: horizontal search windows are used to extract lane markings from intensity images for tracking, see Fig. 1.4, where the lane markers are characterized by dark-bright-dark transitions. After that, an oriented edge detector is used to extract these transitions and form oriented edges, where the lane markings are considered to be the ones in a small interval around the vertical orientation (Dickmanns' algorithm that performs this task is described in [19]). Tracking is established in two phases. First, the algorithm is initialized by extracting the edges that delineate the lane markings from a large search area in the image. Regression is then used to form two lines through the left and right lane markings in the wide-angle image. Straight lines form a good approximation in the wide-angle image because the lane curvature is negligible in the near field (6-30 m) on highways. Using this assumption of a linear road segment in the near field ahead, the vehicle yaw and lateral offset from the road's center line as well as the road width were estimated. Vision was used as the main method of sensing the external environment, but an assortment of other sensors such as accelerometers were used to estimate vehicle motion. This forms the basis for an extended Kalman filter that estimates vehicle and road states along time. Driving was ultimately accomplished by maintaining the lateral offset to the center line of the road, which showed best results without incorporating the vehicle yaw to navigate, [21].

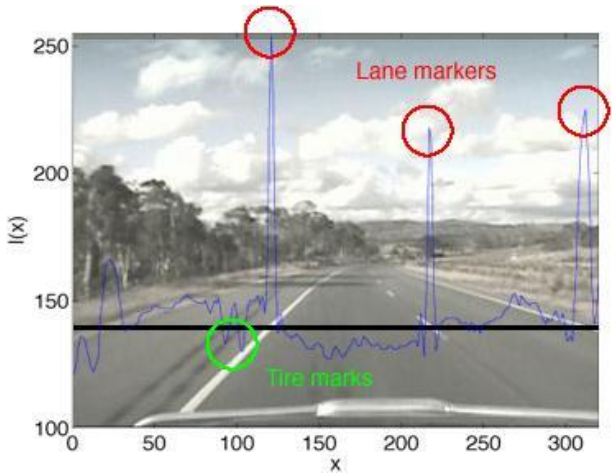


Figure 1.4: Lane marker extraction using dark-light-dark lane profiles. The blue line shows the intensity profile along the black scanline in the image. Lane markers are extracted by looking for dark-bright-dark transitions in the intensity profile (red circles). Other markers such as oils streaks or tire marks (green circle) can also be tracked by searching for representative template profiles, [5].

Also in the scope of the EUREKA Promotheus Project, was the vehicle MOB-LAB (MOBile LABoratory), available to all Italian Research Units involved. When the Promotheus project ended, the research conducted at the Department of Information Technology (Dipartimento di Ingegneria dell'Informazione), University of Parma, continued, funded by the Italian

National Research Council. Within this, the Dipartimento di Ingegneria dell'Informazione acquired a vehicle, a modified Lancia Thema called ARGO, [10], and equipped it with specialized devices for autonomous driving on highways and freeways. The main sensors were two black-and-white low-cost video cameras. In the first week of June 1998, this vehicle was demonstrated to the public and to the research community, thanks to a 2000 km tour throughout Italy, during which, the vehicle drove itself autonomously.

The ARGO vehicle was a predecessor of what can be considered the state of the art for autonomous driving in Europe, and therefore worth of mentioning. It's name is BRAiVE, see Fig. 1.5, and was developed in 2008 by VisLab, the same group as ARGO in Italy. Vislab group (Artificial Vision and Intelligent Systems Laboratory) has developed many test vehicles, transforming most of them into completely autonomous systems. They developed a total of fifteen vehicle prototypes: 1994 - MobLab; 1997 - ARGO; 2001 - RAS (Surface Antartic Robot); 2002 - VW Passat; 2004 - VW Touareg, US Army HMMWV and Tramps 2004 (Vehicle for Grand Challenge 2004); 2005 - Wrecker (Replica of TerraMax during DARPA Grand Challenge 2005), Volvo Truck and TerraMax 2005 (Vehicle for Grand Challenge 2005); 2006 - Vito; 2006 - PReVENT CRF; 2006 - Oshkosh PLS (Autonomous vehicle with the same technology of TerraMax); 2006 - Hummer; 2007 - TerraMax T2 (Vehicle for Urban Challenge 2007); 2008 - Mini (Vehicle equipped by VisLab with a perception system able to record the driver performance, during driving tests under the influence of drugs, alcohol, medicines.); 2008 - Grandeur (Vehicle equipped by VisLab with laserscanner and vision for pedestrian detection in urban areas); 2008 - GDRS Tetravision System; 2009 - BRAiVE.

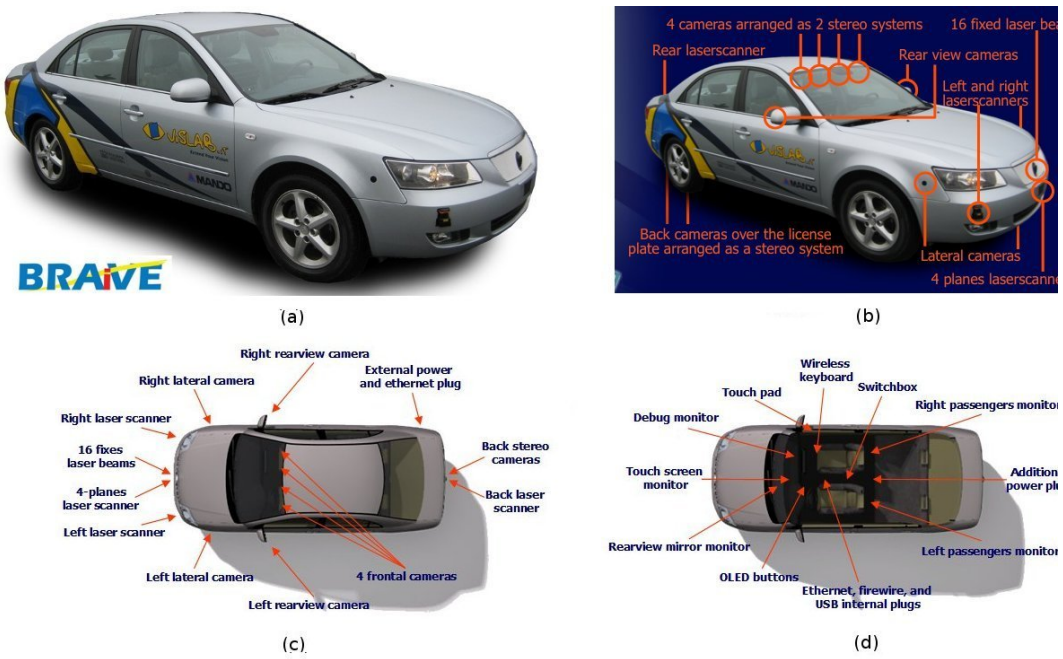
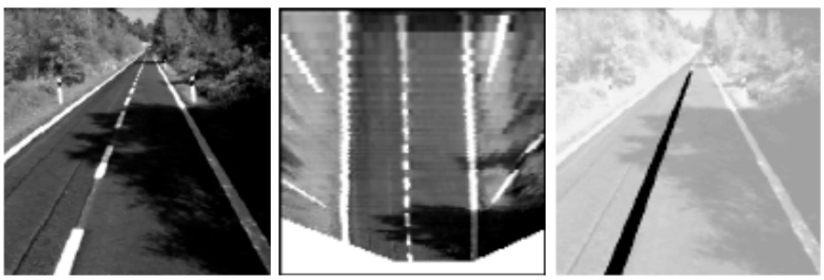


Figure 1.5: BRAiVE (BRAin-drIVE) is the latest vehicle of the well known Italian research group VisLab. This vehicle represents the state of the art in platforms for the study of autonomous driving. (a) BRAiVE vehicle overall look; (b) Sensor suite; (c) External equipment; (d) Internal equipment, [1].

BRAiVE (a short for BRAin-drIVE) is the latest vehicle of the Italian research group VisLab. The car was firstly demonstrated this year (2009), in the first week of June, at the IEEE IV conference in Xi'an, China. BRAiVE is a prototype vehicle entirely developed by VisLab and incorporates many of the systems already created by the group in the last 15 years. The vehicle laboratory is fully equipped with a complete sensor suite and processing systems. All the numerous components are also of the latest technology making BRAiVE one of the best integrated vehicles capable of fully autonomous driving in the world, at least in the current decade. The sensor suite has: a fusion of a total of 10 cameras, 3 single plane laser scanners, one 4-plane laser scanner, 16 laser beams, GPS and IMU (Inertial Measurement Unit).

The General Obstacle and Lane Detection system (GOLD), [8], used since the ARGO vehicle, exploits a flat road assumption to help simplify the problem of lane tracking and obstacle detection. The system transforms stereo image pairs into a common birds eye view using inverse perspective mapping (IPM) under the assumption of a flat road in front of the vehicle. The advantage offered by the use of the IPM is that in the remapped image, see Fig. 1.6, the road markings width is almost invariant within the whole image. This simplifies the following detection steps and allows its implementation with a traditional pattern matching technique, [10]. The lane detection algorithm relies on the representation of road markings by quasi-vertical constant width lines, after the IPM transform, brighter than its surrounding region. Hence, the first step of the road markings detection algorithm is a low-level processing for detecting the pixels with higher brightness value than their horizontal neighbors at a given distance. The following stage of processing is in charge of the reconstruction of the road geometry by building chains of non-zero pixels resulting from the lower processing, [8]. Each chain is then approximated with a polyline made of one or more segments. A road model is used to select the polyline which most likely matches the road's center line. Initially, the vehicle is assumed to be in a specific position (center of the lane) on the road, which at the same time, is assumed to be almost straight. In this situation the road's center line in the remapped image is a straight vertical line that is expected to be found in a circumscribed area of the remapped image. Each computed polyline is matched against this model using several parameters such as distance, parallelism, orientation, and length. The polyline that fits better these required parameters is selected, and represents the center line of the road ahead.



(a) (b) (c)

Figure 1.6: The Gold System. Inverse perspective mapping for lane tracking. (a) The captured road image. (b) The inverse perspective transform of the road image. (c) The detection of the road center line, [8].

Remapping the image into the ground plane and looking for vertical lines inherently enforces the constraint that lane markers are parallel (the vanishing point constraint). The inverse perspective mapping allows for fast image processing algorithms to be used, thus allowing real-time operation. A big constraint on the system is it's limitation to roads with little horizontal curvature and no vertical curvature, [8]. The autonomous steering capabilities result from the image processing (position of obstacles and geometry of the road), used to drive an actuator on the steering wheel. More precisely, the output provided by the GOLD vision system, is the lane center at a given distance ahead of the vehicle. The steering wheel is turned to head the vehicle at that lane center point ahead of the vehicle. In addition, the information coming from a speedometer is integrated to handle changes in the vehicle speed.

In the United States, back in 1980, the Defense Advanced Research Projects Agency (DARPA) funded a robot called Autonomous Land Vehicle (ALV), that achieved the first road-following demonstration that used laser radar (Environmental Research Institute of Michigan), computer vision (VITS system, [51]) and autonomous robotic control (Carnegie Mellon University). The vehicle could go up to 30 Km/h. In 1987 Hughes Research Labs demonstrated the first off-road map and sensor based autonomous navigation on the ALV. The vehicle travelled over 600 m at 3 Km/h on a complex terrain with ravines, large rocks and vegetation. In 1995, the Carnegie Mellon University Navlab project achieved autonomous driving on a 5000 Km trip. This car, however, was semi-autonomous by nature: it used neural networks to control the steering wheel, but throttle and brakes were human-controlled. The Navlab group (NAVigation LABoratory) builds robot cars, trucks, and buses, capable of autonomous driving or driver assistance. Since 1984, they have built a series of 11 vehicles, Navlab 1 through Navlab 11. Their applications have included off-road scouting, autonomous driving in highways, run-off-road collision prevention, and driver assistance for maneuvering in crowded city environments. Their current work involves pedestrian detection, surround sensing, and short range sensing for vehicle control. The latest Navlab 11 is a robot Jeep Wrangler equipped with a wide variety of sensors for short-range and mid-range obstacle detection, see Fig. 1.7. A great deal of info on Navlab methodologies and system architectures can be found on [17].

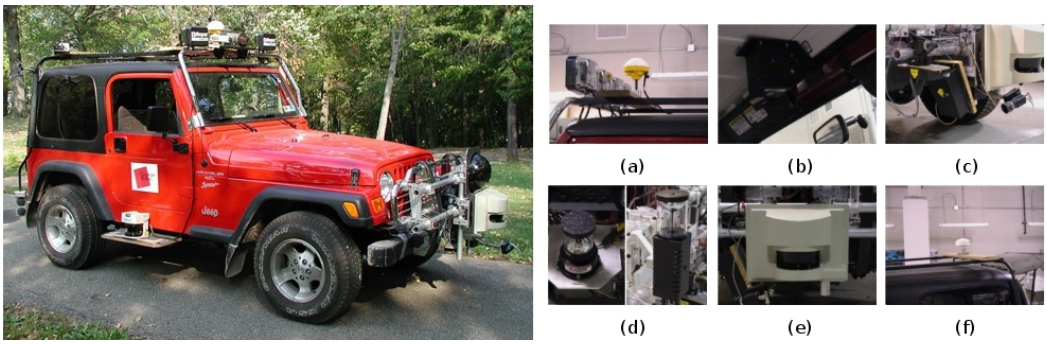


Figure 1.7: Navlab 11 from The Carnegie Mellon University Navigation Laboratory. This vehicle has a wide variety of sensors mounted, some of them being: (a) GPS; (b) Gyroscopes and magneto-meters; (c) Laser line striper; (d) Omnidirectional cameras (x2); (e) Proximity laser scanner; (f) Time & Date GPS antenna.

The Navlab group had not one, but several vision and navigation systems, a more deep reading on these approaches can be found on [49]. The most impressive, is RALPH (rapidly adapting lateral position handler), which operates as follows: the system looks out between 30 and 120 m ahead, and a trapezoid covering this area in the image is sampled. The image inside the trapezoid is geometrically warped to produce an overhead view, a bird's eye view. This transformation causes lane markings on a straight road to become parallel. A one-dimensional vertical histogram of the image intensity in the transformed area is then computed. This histogram has spikes that designate the location of the bright lane markings. For each row of the image, an transformation is applied by adding a value to the x component on each pixel of a row, moving it to the right or left. This is done in order to match the retrieved image with another one that contains parallel lane markings, see image (b), Fig. 1.9.Upcoming road curvature is then computed by comparison of the transformed image with five road curvature hypotheses, see image (c), Fig. 1.9. Template histograms produced by various road curvatures are stored, and compared against the image obtained histogram. The closest match indicates the curvature of the road.

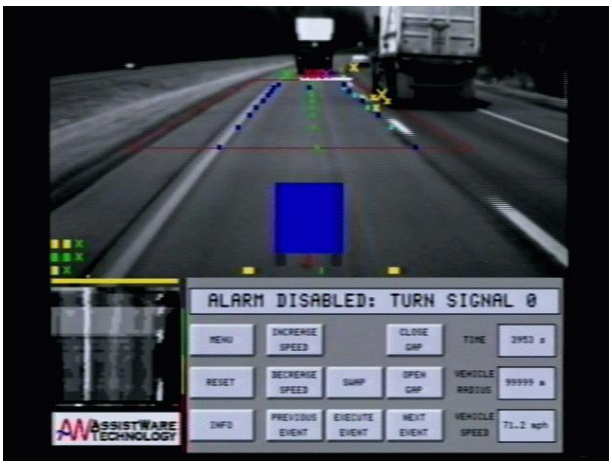


Figure 1.8: RALPH screenshot. The menu in the lower half of the screen is used as part of the automated control system, and allows the driver to change settings such as vehicle speed and following distance, and to initiate actions such as lane changes, [5].

The RALPH interface can be seen in Fig. 1.9, where the trapezoidal area indicates the area being sampled. After the lane markings and curvature are found, they are marked on the interface, see Fig. 1.8, as the dots superimposed over the lane markings. The lighter points between the lane markings indicates where RALPH thinks the center of the lane is. However, RALPH suffered from the inclination to follow departure lanes instead of tracking the lane that it was in. The approach is unique so that isn't restricted to tracking lane markers, but can follow any feature that is parallel with the road. The model is inherently limited to roads with almost no vertical curvature, due to the lack of computational efficiency. The main problems discovered during the trial were associated with poor visibility due to rain, shadows of overpasses and road deterioration, [6]. In addition, the rapidly adapting template introduces the possibility of drift in the estimation. For example, because the far-field is used to adapt the template of the road profile, it's possible for the system to mistaken track features that

are not part of the road, thus letting the tracking to drift off the road. This explains the tendency of the system to follow departure lanes.

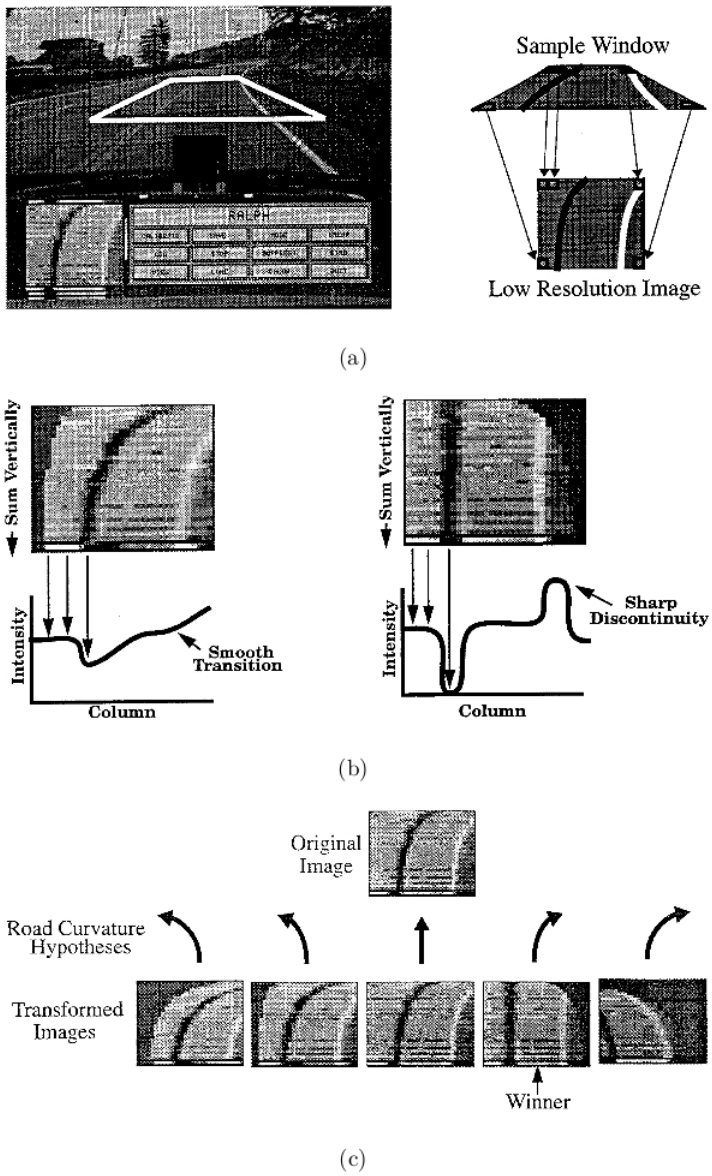


Figure 1.9: Rapidly Adapting Lateral Position Handler (RALPH). A trapezoidal region is extracted from the road image (a) which is transformed to extract the hypothesized road curvature (b) to match the profile in (c), [5].

In 2003 the Defense Advanced Research Projects Agency (DARPA) announced the first Grand Challenge with the goal of developing vehicles capable of autonomously navigating desert trails and roads at high speeds. It's a competition to develop autonomous vehicles capable of safely, reliably, and robustly driving. The competition was generated as a response to a mandate from the United States Congress, The National Defense Authorization Act for Fiscal Year 2001, Public Law 106-398, Congress mandated in Section 220 that "It shall be a

goal of the Armed Forces to achieve the fielding of unmanned, remotely controlled technology such that ... by 2015, one-third of the operational ground combat vehicles are unmanned.". DARPA conducts the challenge program in support of this Congressional mandate with the intent to not only develop technology but also rally the field of robotics, and ignite interest in science and engineering. Organized as competitions with cash prizes awarded to the winners ($1 million in 2004; $2 million in 2005; $2 million for first, $1 million for second, and $500,000 for third in 2007), as an alternative to the conventionally funded research. For the first two years the government provided no research funding to teams, forcing them to be resourceful in funding their endeavours. For the 2007 Urban Challenge, some research funding was provided to teams through research contracts. By providing limited funding, the competition encouraged teams to find corporate sponsorship, planting the seeds for strong future relationships between academy (the source of many of the teams) and industry. The competition is open to teams and organizations from around the world, as long as there is at least one U.S. citizen on the roster.

The 2004 Grand Challenge was framed as a cross-country race where vehicles had to drive a prescribed route in the minimum time. There would be no obstacles on the closed course. It occurred in the Mojave Desert region of the U.S.. But the 228.5 Km course proved to be an impossible mission for the competitors, with the best vehicle traveling only 11.84 Km. In the 2005 Grand Challenge, all but one of the 23 finalists achieved the 11.84 Km distance completed by the best vehicle in the 2004 race. A total of five vehicles successfully completed the race. The third competition of the DARPA Challenge, known as the 2007 Urban Challenge, took place in a isolated urban environment. The course involved a 96 Km (60-mile) urban area course. Rules included obeying all traffic regulations (now the cars have to deal with road intersections), while negotiating with possible obstacles (like other competitor's vehicles). Although the 2004 and 2005 events were more physically challenging for the vehicles, the robots operated in isolation. The Urban Challenge required designers to build vehicles able to deal with intersections and avoid other vehicles on the course. In all three DARPA events, tunnels were also included on the route to induct lightning variations. Six teams successfully finished the entire course, all in an average of 20 Km/h. It is far to notice that the DARPA cars heavily use GPS, always driving from one way-point to the next. The DARPA course is unrehearsed by the teams but precisely given by almost 3000 known way-points, with several way-points per curve.

1.3 Experimental Platform

The concept behind the used study platform is the idea of reducing the scale of a real typical driving situation to a smaller simulation environment. This is very useful because despite of being a great topic, not everyone has the resources to acquire a car and transform it into an autonomous vehicle, in order to conduct research on the autonomous driving field. The resources to build an experimental platform to support this type of study are very expensive. So, in order to reduce expenses and limitations, we have a reduced simulation environment with reduced cars, where we can simulate real driving situations with artifacts like traffic lights or obstacles, road maintenance areas, crosswalks, etc, without affecting the general aspects and requirements of the initial problem. The idea is to develop systems and

techniques that can be later applied to autonomous driving vehicles.

The Festival Nacional de Robótica (Portuguese Robotics Open) committee promoted this conceptual environment by the first time when the event held its first edition in 2001. The event aims at promoting science and technology through robot competitions and takes place every year in a different city, [2]. It also includes scientific meetings where researchers in the area of robotics come together to present the latest results of their activity. This event has had, since its beginning, a tremendous growth, both on number of participants and teams, as in terms of audience. It includes various leagues of competition, in particular, the Autonomous Driving Competition, as well as the ones that follow the RoboCup official rules: Soccer Middle-Size League, Junior Soccer Leagues, Rescue Junior and Dance. The Autonomous Driving Competition, of main interest to this work, takes place in an indoor structured area that tries to reproduce, to some extent, a real life scenario. The environment is represented in Fig. 1.10.

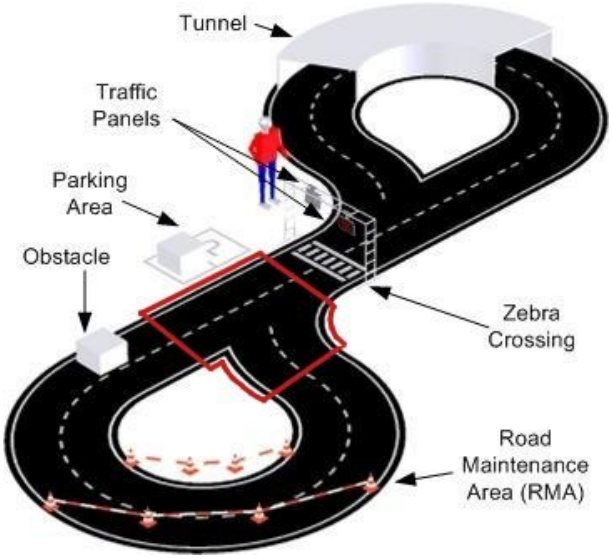


Figure 1.10: Simulation environment.

The path used, [7], has a 8-shape configuration delimited with two white lines to simulate a two-way road, see Fig. 1.10. Also has the following additional elements: a crosswalk with a pair of traffic light panels (one on each direction), a tunnel, a road maintenance area, an obstacle to be placed in an unknown position and a parking area. The entire 8-track, when assembled, has a total size of 14 x 9 m. The possible traffic signs used on the environment are shown in Fig. 1.11, and are: "Stop", "Go Forward", "Go Right","Go Left", "Lap Finish". This simulation environment has been used in the scope of the Portuguese Robotics Open, aiming at promoting multi-discipline technical developments towards autonomous driving.

In this simulation environment, many important aspects addressing real driving situations can be replicated. The environment has a lot of potentialities: in it we have the possibility to simulate a two-way track (delimited with two continuous lines and one discontinuous between them) for tasks like road following while keeping the car within the right lane, lane changing,

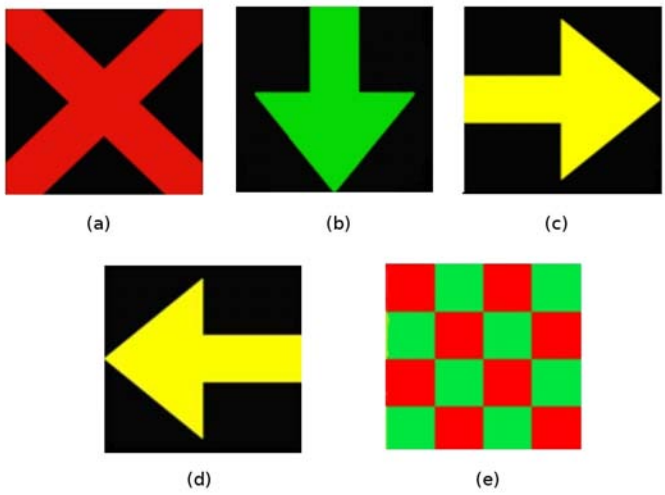


Figure 1.11: The five traffic signs used: (a) Stop; (b) Go Forward; (c) Go Right; (d) Go Left; (e) Lap Finish.

evasion maneuvers (e.g. avoid obstacles). It also contemplates road signs identification and the necessary situation awareness decision, or movement and parking in the parking area. There is also a special bifurcation scenario (in red in Fig. 1.10) in which two road structures converge and become one, or diverge (if you look at it from the opposite direction), laying here the more complex task of identifying the two directions the car has in front of it, and making the decision on which direction to take, within a safe route. Also, thinking of the track as a free race track, other tasks could be to determine the correct and shortest route and ran it in the shortest time. A more complex task consists on mapping the track and locating the car on it. As an extra to the already existing environment, in the future there could also be included moving entities (simulating e.g. cars or persons), this would involve tasks like building dynamic models of them and would require other decision layers. As the environment resembles a conventional real urban environment, it is expected and assumed that all the work done in this environment can be adapted and re-engineered to work in real situations.

ROTA, see Fig. 1.12, is the three-wheel robot car manufactured at University of Aveiro, designed and implemented from scratch by graduate students, in the scope of their final year projects. The robot, [7], powered by two 12V 7Ah batteries, has a electric traction motor connected to the rear wheel that can provide speeds up to 2.5 m/s (9 Km/h), which is fast enough for the car's reduced scaled environment. The car moves in both directions (forward/backward) and the motor also has the ability to act as a brake. For the steering, a simple proportional servo motor is connected to the two independent front wheels, allowing the curving of the robot. The robot also has headlights, brake warning light and left and right blinkers. The low-level hardware layer has a set of nodes interconnected by means of a CAN (Controller Area Network). This network is complemented with the FFT-CAN protocol (Flexible Time-Triggered communication over CAN) for improved performance, [7]. A gateway interconnects the CAN network to the PC at the high level, providing the control needed over the car components, see Fig. 1.13. At the high-level coordination layer, the main processing unit is a PC based computer running Linux, with enough interfaces to accommo-

- Dimensions (length x width x height) $\simeq$ 51 x 33 x 28 cm
- Maximum velocity $\simeq$ 9 km/h
- Wheelbase distance $\simeq$ 53 cm
- Maximum curvature $\simeq$ 1 m^{-1}

Figure 1.12: ROTA - RObot Triciclo de Aveiro and static properties.

date the two FireWire cameras mounted on the vehicle, and the necessary communication with the low-level layer (serial/USB port). The specifications of the car are described in Fig. 1.12.

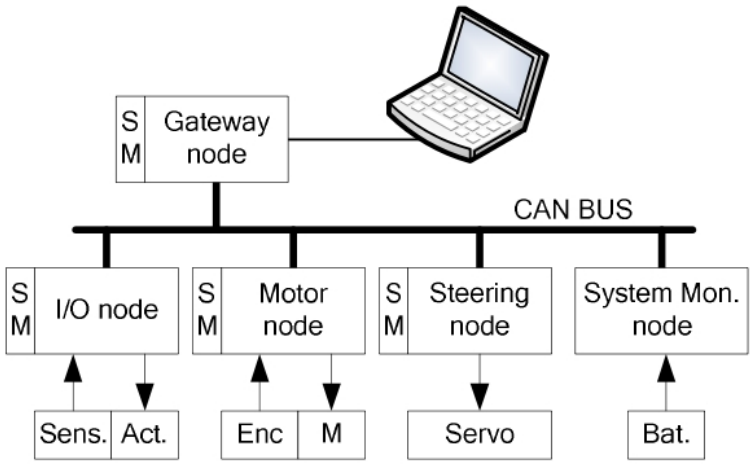


Figure 1.13: ROTA low-level hardware architecture, [7].

The ROTA's vision module uses two FireWire (IEEE 1394) cameras, see Fig. 1.14, connected directly to the computational system. Both cameras are set on top of the robot. One of them is used only for the traffic lights detection, mounted on the rear of the car, Fig. 1.15. This placement is such that the robot is able to see the traffic lights panel, when stopped at the crosswalk. This would not happen if the camera was mounted in the front of the robot. The other camera, see Fig. 1.16, is mounted on the front of the robot and is directed downwards. It's used to retrieve the lane, obstacles, road maintenance area, crosswalk. The camera is used to detect every element except for the traffic lights panel. Both are low-cost cameras, presenting some intrinsic limitations and have a mix of fish-eye and radial distortion.

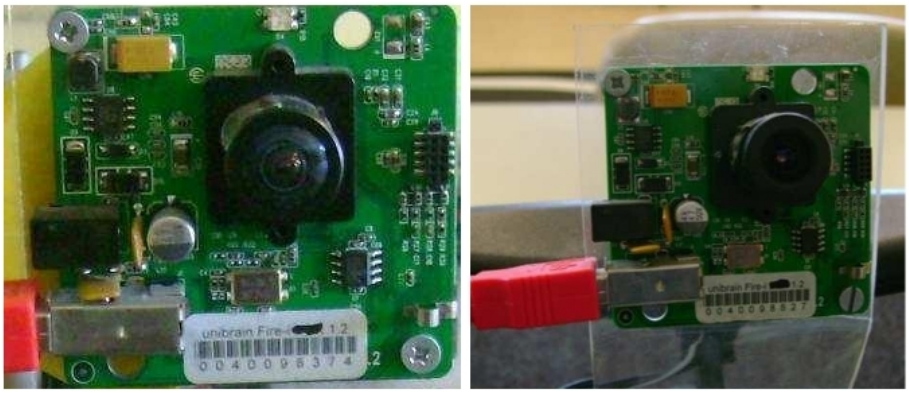


Figure 1.14: The Firewire cameras used in ROTA's vision module.

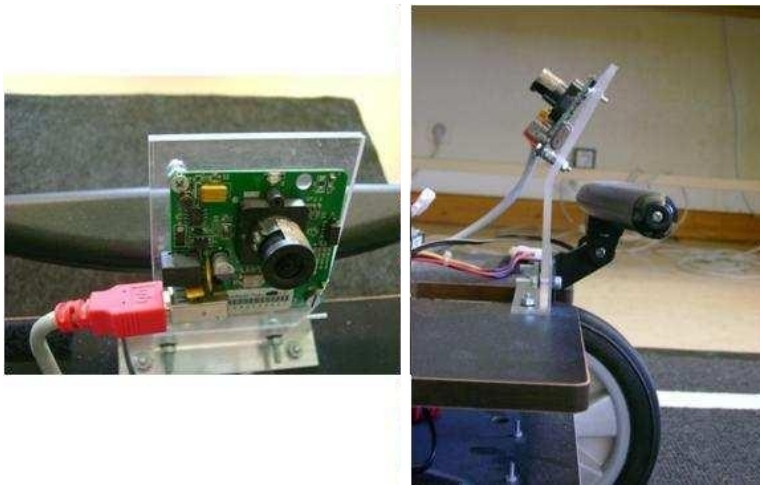


Figure 1.15: The rear camera used in ROTA, for the traffic panels detection.

The cameras support different acquisition formats and their frame rate can be configured. This is important to balance computation performance requirements with the frame rate acquisition. A higher frame acquisition rate is important, so that the robot is aware of what is happening around it, and achieve correct and smooth driving at top speeds. For example, if the frame rate is 4 fps and the robot drives at a low speed, 0.5 m/s, the robot will travel 12.5 cm between the frame acquisitions, without information about possible changes in the environment. At this speed, this can prove to be redundant, but at higher speeds, this acquisition rate isn't sufficient for proper navigation. Also, the intention is to detect any obstacle as soon as possible, so that the proper behavior can be performed. The chosen acquisition format for the cameras is the YUV422 with a resolution of 320 x 240 pixels, since it's the one that allows for faster acquisition rates. Although an average of 15 fps rate is sufficient, the intention is to always maximize the frame rate, balancing it with the computational costs of the robot's software.

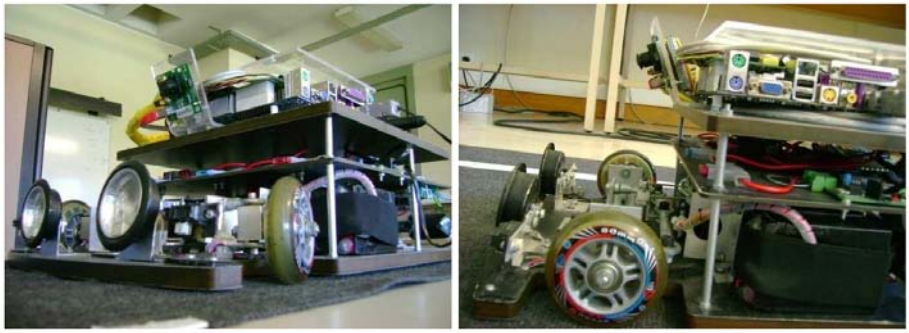


Figure 1.16: The front camera used in ROTA.

1.4 Previous Work

The major part of the control models used in the autonomous robots that participate in the Autonomous Driving Competition (at the Portuguese Robotics Open), all have in common that are based essentially on state machines and classic control to follow the white lines in the 8-path. The autonomous systems developed outside this scope, in fact, are also modeled within similar approaches, since this is the most traditional and easiest approach to achieve a solution for autonomous driving. The different states represent behaviors that the robots must perform to complete the competition challenges. The connection between the different behaviors is intimately tied to the transition between different sections of the path. So that the transition between behaviors is usually done when there is a transition between sections, on the occurrence of a lane change (the path is two-way delimited by a discontinuous lane marking). In the crosswalk section, the behavior transition is triggered by the sign showed on the traffic lights panel. Since the scenario is previously known, the approach to the crosswalk can be triggered by the distance traveled since the starting point, the crosswalk (from the starting point, by performing a half-lap the robot reaches the crosswalk again in about 21 meters).

The state machine that achieves the desired control of the ROTA robot, is represented in Fig. 1.17, [18]. Each rectangle represents a state and the arrows represent the conditions for the state transition. At each state the robot performs specific tasks and the initial placement of the robot is at the crosswalk, where after waiting for the traffic lights display, the robot will leave it and execute one of three possible behaviors:

- *FollowLine*- if the traffic sign displayed is a green upper arrow, then the robot will go forward and continue along the current lane.
- *LeftTurn* - if the traffic sign displayed is a yellow left arrow, then the robot will execute a left turn, and then go along, keeping inside the current lane.
- *Park* - if the traffic sign orders to go right, which happens when the number of half laps is 4, it means that the robot has concluded the laps required and should attempt to park.

When in the crosswalk, the robot's state *WaitingforSignal* will change accordingly to the traffic lights panel. When the sign changes the robot can perform the next *FollowLine* be-

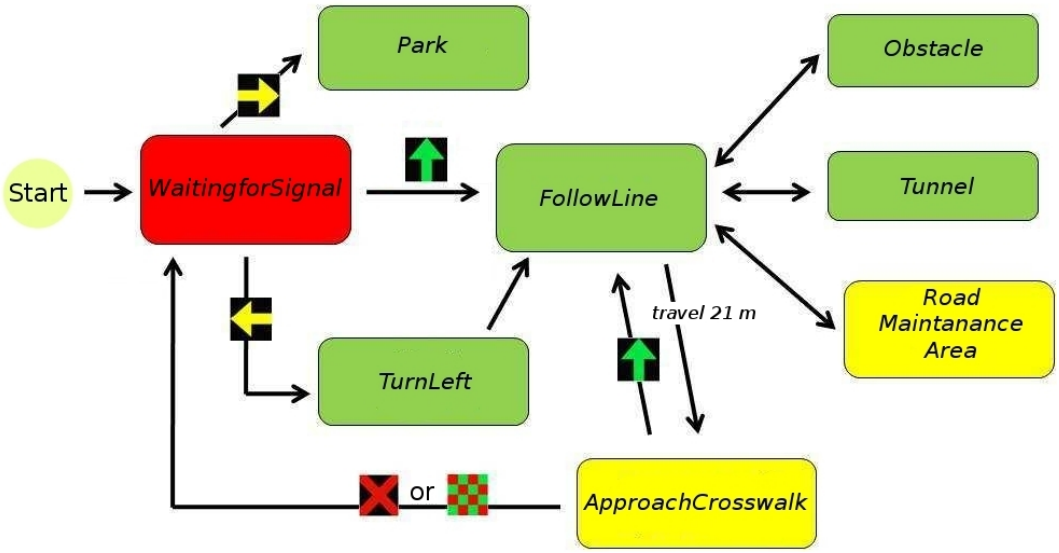


Figure 1.17: The different behaviors and condition transitions, that achieve the desired control of the ROTA robot, [18].

havior by going forward, as displayed in the traffic sign, and traveling a half-lap around the track in the counter-clockwise direction. Or can perform the *LeftTurn* behavior, that consists on executing a turn left maneuver ending with a change to the *FollowLine* state, [18]. In this machine state, the robot essentially occupies most of it's time on the *FollowLine* behavior, and does what actually almost every approaches inside the Portuguese autonomous driving context do: follow the white lines and keep inside the current way lane (left or right, or consider the both as only one, this is particularly done by the bigger robots). The other states are created so that the robot can deal with the traffic signs and be able to stop when the half-laps are completed, which only happens in a crosswalk proximity. So the quality of the autonomous driving of a robot is defined almost exclusively by the ability to keep between the two white continuous lines. In the Portuguese Robotics Open, less than 50% of the participants can accomplish this task and finish the 4 half-laps, which shows the difficulty associated with this apparently simple operation.

The image algorithms used are solely based on color segmentation and image sensors, in order to find the occurrence of specific colors, as represented in Fig. 1.18. The lines detection is illustrated in the image by horizontal lines, representing the image sensors, and the resulting detection points, used to drive the car. The lane markings detection is limited to the nearest ones, and the number of image sensors used is static. By using the detected position of the white line with the horizontal search, the robot can move along a lane by correcting the angle in the directional front wheels, so that it minimizes the deviation between the detected line point coordinates and the coordinates where the line should be, if the robot car was aligned with the current lane direction. So if the detected line, in Fig. 1.18, is too much to the left, the robot will tend to go left, if the same is to the right, the robot will go right. By this, the robot car corrects its travel path by proportional control, such that the corrected angle c_a, is

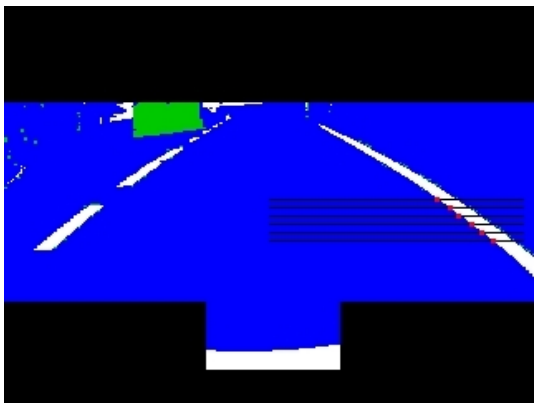


Figure 1.18: Image sensors used in ROTA vision system. The black areas are used as a static mask, meaning that they have no relevant information, preventing the segmentation process from processing them, [18].

the actuation applied to the directional front wheels and is given by:

$$c_a = k * (pt_{detected} - pt_{desired}) \tag{1.1}$$

where k is a constant that smooths the reactive path, and is determined by experimental tests. The line point current detection coordinates and the desired ones are respectively $pt_{detected}$ and $pt_{desired}$. For $c_a < 0$, the front wheels will make the robot go left, and for $c_a > 0$, go right, correcting it's alignment with the current lane. The vision system can maintain the car within the lane, and detect: the obstacle, crosswalk and traffic lights panel, based on color and the image sensors. Still, the lane detection limits the maximum velocity that the car can perform, and there are not methods to solve the road maintenance area.

1.5 Motivation

The software layer architecture that was previously developed for ROTA, was specifically conceived to solve the 8-path scenario challenge described, and participate at the Portuguese Robotics Open. The initial point of the robot car has to be always specific, for the correct architecture flow, see Fig. 1.17, which resides on the beginning of the crosswalk section. The robot also does not possess any method to perform global searches, so it can't be randomly dropped in the track and successfully achieve a lane detection and subsequent knowledge of its own position relative to the road. This means that it can't determine whether it is on the right lane or the left one. Furthermore, the state machine requires an initially known environment state for the car, i.e., the car needs to know where it stands beforehand, which is due to the fact that the agent was designed explicitly for the autonomous driving competition, where the initial position is always at the crosswalk. The computer vision methods used for the environment detections are based only on the data incoming from a color segmentation algorithm. These computer vision methods require a time consuming calibration process and are very sensitive to changes in illumination, since this affects how the colors of the environment are perceived through the cameras.

On the scope of this thesis, the motivation is to design a complete new perception layer, introducing new methodologies and providing a more accurate and efficient vision system. This perception layer should autonomously extract and maintain an informational state about the world. The perception is to be independent from the behavioral layer, focusing in the maintenance of the best representation for the state of the world. When a representation is acquired independently, an artificial intelligent agent can be more easily implemented to use the maintained knowledge about the world, in order to achieve some goals.

By using a model-oriented software approach, truth maintenance tables and perceptual inference rules, this perception layer is to be designed and implemented from scratch, i.e., without the use of the current implemented architecture. The new perception layer is to be independent from the behavior layers, self-adapting and provide more reusability. A major motivation is to develop a software layer that can be used not only in ROTA and simulation environments, but without any changes, in real world scenarios, within real cars, in complex dynamic environments, which is, the nature of the autonomous driving context. This can be extremely difficult, particularly when the solution is essentially based on computer vision applied to camera's images. Within this thesis scope, the environment seen through the cameras, is inherent to be at all time, partially observed, as so is the information extracted from it.

1.6 Goals

The scope of this thesis limits itself to the study of perception and localization for autonomous driving. The actuation layer developments are part of some other parallel projects, therefore, this thesis involvement in such context is always kept to a minimum. The first goal is to develop a Vision Perception System that can replace the current one, and accomplish all the necessary detections successfully. It is intended that the search for lane markings uses a major portion of the image, so that all the visible image points of the lane markings, within the image, are taken into account for a better representation of the lane. Since more points are gathered, the information about the lane markings is wealthier, and a car pose can be achieved. The perception layer should attempt to maximize its own utility, by gathering the maximum number of image points and focusing on keeping the needed computational time cost to a minimum. This can be done by minimizing the image processing by selecting regions of interest in the image, or actions of interest. Regions of interest are specific areas of the image where the algorithms are to be performed. An action of interest can be to do a more exhaustive global search for a specific lane marking, or other element, at some moment in time. The car should be able to determine where it is and the current state of all the possible environment elements, at all times, even if dropped randomly in any place on the track.

The idea is to create a completely independent Perception layer, responsible for maintaining knowledge about the world state, choosing the best way to deal with the information that is retrieved from sensors, and the one that it already has. This is procedural, in the way that each search and focus areas on the image are chosen based on the current knowledge about the world state, and after that, the feedback is used to deliberate a validation on the world model and maximize the layer's utility (selection of regions of interest). The aspects just described consist on the base of the Perception Reasoning System, which development is

the primary goal of this thesis. It's features are:

- an independent perception layer;
- autonomous in the task of interpreting the world state;
- focus of perceptual attention, using this to maximize the layer's utility and computational speed;
- evolve to the next world state based on the previous ones;
- reactive and model-oriented behavior with modeling delayed feedback;
- handling of incomplete and/or inaccurate data.

These aspects are the basis of a Procedural Reasoning System (PRS), which is an Artificial Intelligence framework, for constructing real-time reasoning systems that can perform complex tasks in dynamic environments, [27]. It is based on the notion of a Rational agent or Intelligent agent.

The complete implementation of the Vision Perception layer that use cameras for retrieving the current visible environment, is the primary goal. But, when concluded, it can be used to achieve more ambitious goals. The approaches that are here considered to achieve autonomous driving are:

driving by road ahead uses only the current perception of the observable road to determine a target point and consequently a trajectory to follow. This, can be done using the classic control method, already described in the previous work.

local-pose driving uses the observable road ahead to estimate, by means of extrapolation, the position of the car geometric center, hereby the middle of its rear wheel. With that, achieve short-term planning. Here, the angle of the car's initial position and the one on the target position are considered for the trajectory, resulting in an improved and more smooth planned path.

world-pose driving uses the local-pose estimation and a well defined mapped track, to estimate a world-pose as a global positioning system (but without any GPS measures). The car is always related to a track already mapped and makes use of the car dynamics analysis and the vision component to estimate the global position. With this, the robot achieves long-term planning, and environment mapping. This is useful at several levels, for example, in the 8-path, the elements that are already seen on a first lap, can be used in long-term path planning, so that the proper actions to deal with this element can be taken much sooner. The robot can also use this information to plan, for example, a shortest route.

For each driving approach, a different level of perception abstraction is needed. So that the driving by road ahead only requires the road ahead representation for defining a target point. For the local-pose driving, the robot needs also to know about it's current pose related to the road ahead. For the world-pose driving, a mapping component is also needed, and the use of it with the local car pose and dynamics allows the robot to locate itself in a world map (self localization), and therefore, long term intelligent planning. For each, there is also a different type of behavior associated, being the driving by road ahead the most reactive one,

and the world-pose driving the most deliberative. The goal is to achieve these abstractions, so that they can be used by an Action layer.

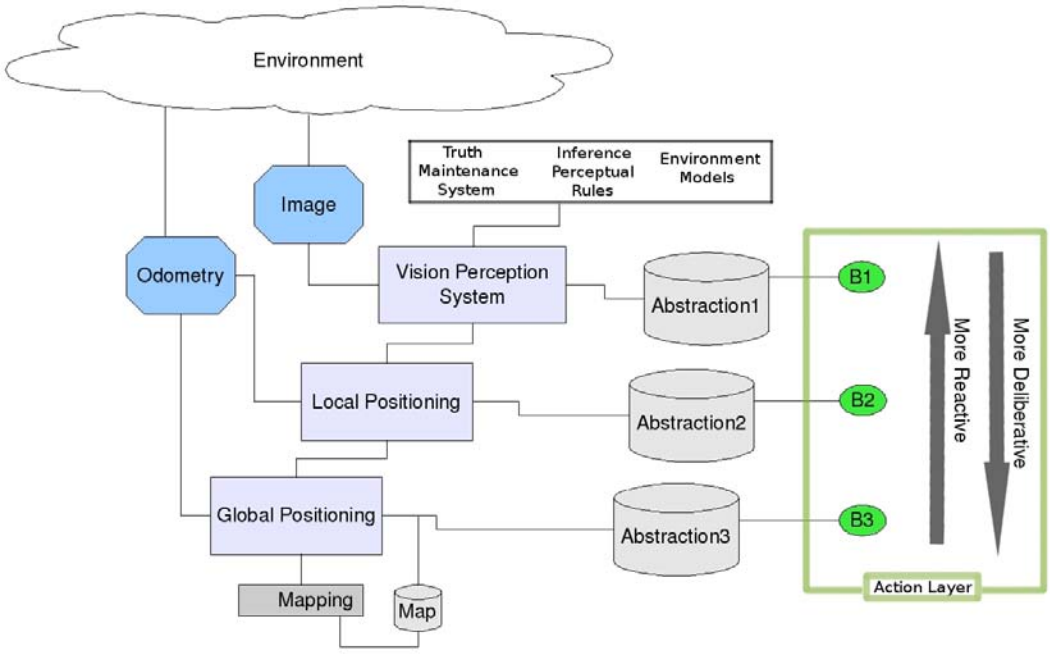


Figure 1.19: The entire proposed Perception layer design. The Action layer, which conception is outside this thesis scope, should use one of the three abstractions provided. The software agent can then perform one behavior $B1$, $B2$ or $B3$, based on some internal logic.

The goal is to first develop a Vision Perception System, achieving all vision tasks required in ROTA, and afterwards it can be extended to achieve self-localization, by determining the local-pose of the robot car relative to the road ahead and later the world-pose, using the local-pose and mapping to achieve global positioning. The proposed architecture for the Perceptual Layer is represented on Fig. 1.19, where the vision-based perception system updates the environment state using perceptual inference rules, a truth maintenance system and environment models. The various behaviors $B1$, $B2$, $B3$, can be performed by some type of software agent based on some internal logic. The advantage is that the Action layer does not have to deal with any perception issues, and can be programmed independently. The robot car can be driven based on one of the three layers of abstraction, however, there is the possibility to co-exist different behaviors and is up to the software agent to decide on which to perform at some point in time.

1.7 Achievements

It was designed and implemented a new Vision Perception System, capable of detecting with large success, the road ahead of the car, without any calibration required. The system also has a bigger immunity to environment variances and can detect lane markings of any

color, not just white ones. This is achieved by using the Canny algorithm and a Reasoning System. It uses the feedback, classification and model validation for perceiving the environment in minimum processing. It is also successful for the cases where the robot is completely lost or is far from being aligned with the current lane. This lane detection system proved, by experimental testings, that it can supply the car with the ability to drive at its maximum velocity without any issues. It was developed a method that can approach what type of section is in front of the car, and supply an error approximation for the lane detection system, as for an error for each lane marking so that they can contribute independently to the lane result. This lane error estimation can result in a confidence on how much the robot believes in the current perceived environment.

It were also developed a set of successful methods for the traffic lights panel detection, the obstacle detection and the road maintenance area. The development of the IPM module provides all the detected environment elements in real distances through a "pixel to real" look-up table, with the proper fitted lane markings, result of regression's analysis.

For the self localization, were designed and implemented methods that allow the determination of the car pose for a local scenario. For the global scenario, methods were developed to estimate a global car pose by computer vision, odometry and mapping with a extended Kalman filter.

The overall goal, defined to be the achievement of methods for the abstraction perception layers described in the motivations, was accomplished. With that, the deep study of a vast science like Robotics and Computer Vision, as for the specific study of the optimal Kalman filter, in which a considerable amount of time was spent, are also considered great achievements that were accomplished.

1.8 Tools Used

The tools used in the work supporting this thesis were: *Git* - a fast version control system, [3], used for collaboration and version control. It is new in the ROTA project, and was used for the workflow of parallel projects; *Gitk* - for graphic version control. This is an interface program to *Git*, which resulting screenshots can be seen in Fig. 1.20; *Doxygen* - a documentation generator, [4], used to document all the implementation work. Some screen captures of the resulting documentation are found in Fig. 1.21, the documentation tools provides a lot of useful features: it can automatically generate useful dependency and class diagrams, as for documentation on-line in *HTML* or *UNIX* man pages, and off-line in *LaTeX*, and consequently, *PDF*; *GCC* - the *GNU* compiler collection used for all the code written in C/C++; *Ubuntu*, *Linux* - v.8.04 - *Hardy Heron* as the operating system; *libdc1394* - library used as API for the acquisition of the *IEEE 1394* based camera's image; *OpenCV* - *Intel*'s library used for some computer vision processing and visual debugging; *gnuplot* - program that is used for the scatterplot of results.

This thesis was written in *LaTeX*.

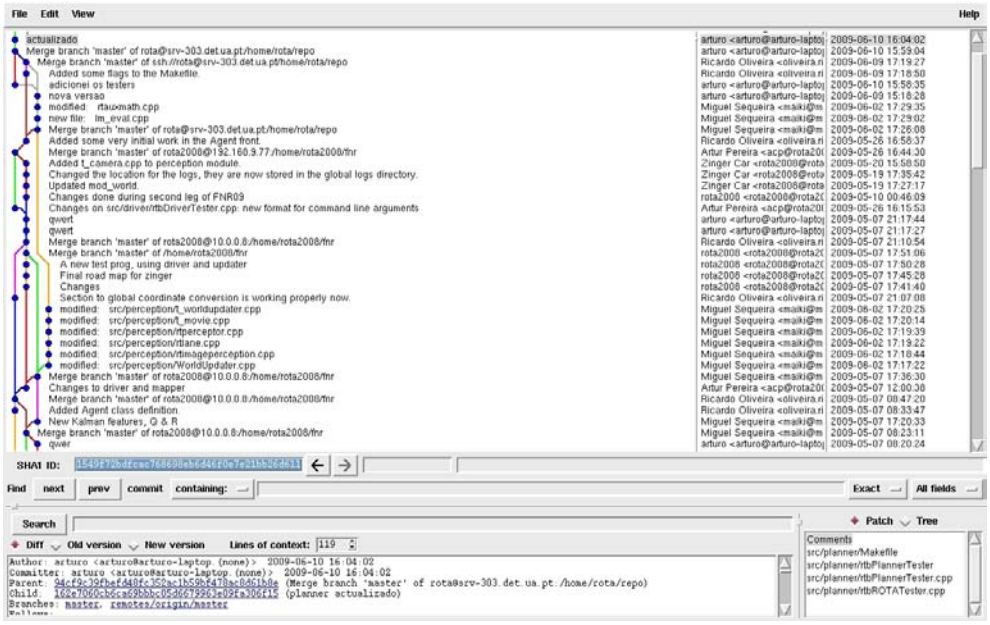


Figure 1.20: In the figure, a screen capture of *Gitk*, a IDE used to work with the *Git* version control system.

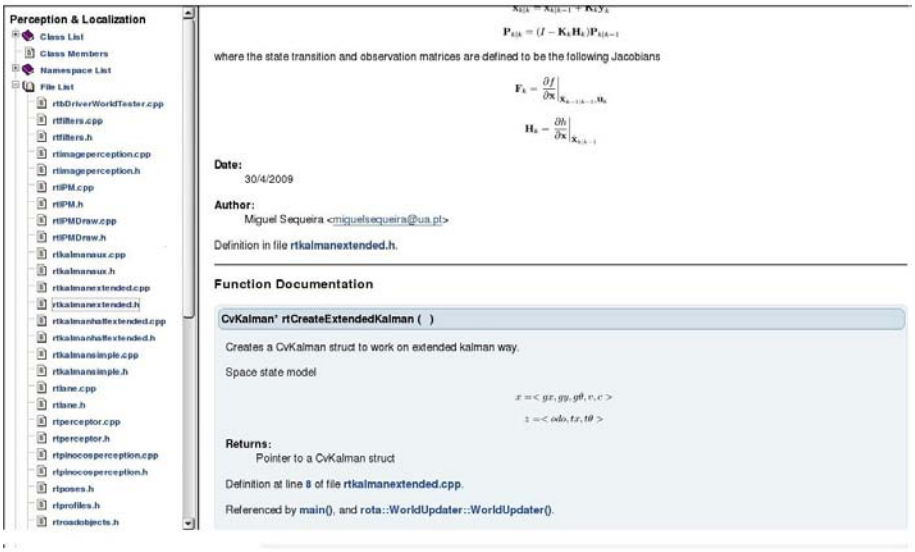


Figure 1.21: A screen captures of the documentation for the project using Doxygen.

1.9 Thesis Structure

The thesis is composed of the following chapters:

Chapter 1 is a general approach to the autonomous driving context, including the history and state of the art. The experimental platform is described, as well as the previous work. The chapter ends with the thesis motivations and goals.

Chapter 2 has descriptions on all the image feature extraction low-level methods, that are used in the Vision System addressed in chapter 4.

Chapter 3 is where the fusion and estimation of information is addressed. The methods here described are used at several abstraction layers. They are used in chapter 4 by the Vision System and also in chapter 5 for Self Localization purposes.

Chapter 4 contains the description of the Vision Perception System. The detection methods for the environment elements are presented, and the chapter ends with a method designed to translate all the gathered information to real world coordinates.

Chapter 5 presents the developments made towards Self Localization, developed so that the robot can drive at the highest level of abstraction.

Chapter 6 is the final part of this thesis and contains the conclusions of the developed work and some suggestions on possible future work.

Chapter 2

Vision Feature Extraction - Low Level Processing

In this chapter some common techniques used to do image low level processing are presented. Techniques like color segmentation, edge segmentation, image morphology, and template matching, that were used during the work supporting this thesis, are described in the next sections. During this introduction will be described the analogy between the human vision process and the computer vision methods used in this thesis work, followed by the overall requirements and difficulties of the feature extraction process.

Let's think of the way we drive our car and perceive the environment. We spot a red car on the side of the road using color, recognize a lane boundary by a distinct transition from the road color to the land marks, and a road sign by its shape. In a similar fashion to our human vision system, it was used color, shape and context to perceive an object in the environment. Some methods like color and edges segmentation, image morphology and template matching (all described in this chapter) were used to achieve the detection of such features.

The need for low level feature detection pre-processing arises when we have a too large set of data and there is the need to simplify the analysis by drastically reducing the amount of data to be processed, while at the same time preserving useful structural information. In our case, for example, we have a new image with 320 x 240 pixels, each with three channels of one byte each, every 33 ms (on a 30 fps acquisition rate), and the pixel components by themselves are redundant information and insufficient for our detection purposes Mostly we also have to consider a generalized neighbourhood around a pixel to decide if its part of a feature or not. So we need to have a way to create an abstraction of the image information by making local decisions at image points, to understand if that point relates to an image feature. In this way we have resulting features to be used as raw material for our detection and tracking algorithms.

In our context, there are obvious time constraints to image processing and feature detection, that can become very computationally expensive, so all our work is directed to be as much as it can, region-of-interest oriented, reducing computational cost. So the algorithms can operate not only on entire images but also on image areas. Often abbreviated as ROI, an image region of interest is a rectangular area that may be either some part of the image or the whole image. ROI of an image is defined by its size and offset from the image origin

as shown in Fig. 2.1. The origin of an image is always implied to be in the top left corner, with x values increasing from left to right and y values increasing downwards. The ROI is a selected subset of our image data identified for a particular purpose (e.g. right lane marking detection area). In this way, the feature detection is guided so that only certain parts of the image are processed. This guidance is later done by the higher level detection algorithms in the high-level perception layer.

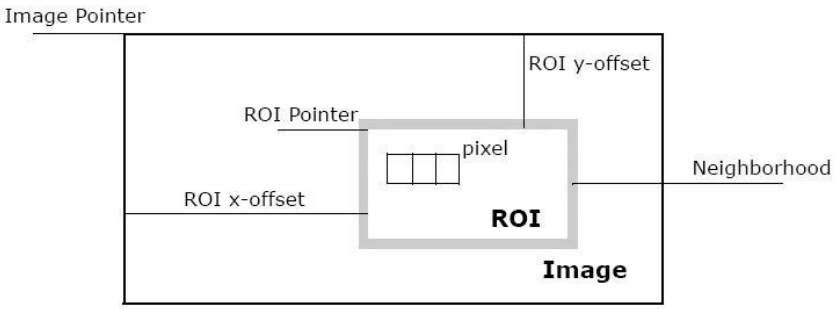


Figure 2.1: Region of interest - all methods in this work are designed to support this useful approach that consists in defining a rectangular area that may be some part of the image for reduced processing.

In this chapter are presented and described the vision feature extraction algorithms introduced in the ROTA project. All them are new, in the sense that they were never approached or implemented before, with the exception of Color Segmentation, which was the source for features and vision information used in ROTA until now, where all algorithms relied on specific pre-defined colors in the environment, resulting in poor computer vision algorithms and versatility. The vision algorithms were also extremely dependent on light conditions and calibrations. The Color Segmentation implementation had design flaws that were now corrected and its functionality was extended. The new methods provide new features, that result in perception versatility and a more extensive and robust platform for new incoming approaches in ROTA.

2.1 Edge Segmentation

Edge detection has been an essential part of many computer vision systems, as a problem of fundamental importance in image analysis. In typical images, edges characterize object boundaries and are therefore useful for segmentation, registration, and identification of objects in a scene. In practice, edges are usually defined as sets of points in the image which have a strong gradient magnitude or where the image brightness changes sharply leading to discontinuities (significant variations of the grey level image). It is also good to keep in mind that in our context is used a two-dimensional image to represent a three-dimensional scene, so edges typically reflect inherent properties of the three-dimensional objects. They can represent surface markings and surface shapes but also discontinuities in depth or changes in material properties. Edge detectors have been many times used on autonomous driving in the past. Some examples can be found on [6], [26], [14], [28], [32], [31]. The result of the edge

segmentation process is shown in Fig. 2.2.

Figure 2.2: Demonstration of the Canny algorithm applied to an image taken on the course Aveiro-Viseu (A25), representing an outdoor real situation, therefore with no control on the light conditions.

2.1.1 Edge Detectors Methods

For a continuous image $f(x, y)$, where x and y are the row and column coordinates respectively, we typically consider the two directional derivatives $\delta_x f(x, y)$ and $\delta_y f(x, y)$. Of particular interest in edge detection are two functions that can be expressed in terms of these directional derivatives: the gradient magnitude and the gradient orientation, as defined in [34]. The gradient magnitude is defined as:

$$| \bigtriangledown f(x, y)| = \sqrt{(\delta_x f(x, y))^2 + (\delta_y f(x, y))^2} \tag{2.1}$$

and the gradient orientation is given by:

$$\angle \bigtriangledown f(x, y) = \arctan\left(\delta_y f(x, y)/\delta_x f(x, y)\right) \tag{2.2}$$

Local maxima of the gradient magnitude identify edges in $f(x, y)$. When the first derivative achieves a maximum, the second derivative is zero. For this reason, an alternative edge detection strategy is to locate zeros of the second derivatives of $f(x, y)$. The differential operator used in these so called zero-crossing edge detectors is the Laplacian operator which is a second order differential operator in the n-dimensional Euclidean space.

$$\bigtriangledown f(x, y) = \delta_{(x,2)} f(x, y) + \delta_{(y,2)} f(x, y)) \tag{2.3}$$

These are some common edge detectors developed in the past: Roberts Cross Edge Detector - 2x2 gradient edge detector; Sobel Edge Detector - 3x3 gradient edge detector; Canny Edge Detector - non-maximal suppression of local gradient magnitude; Compass Edge Detector - 3x3 gradient edge detectors; Zero Crossing Detector - edge detector using the Laplacian of Gaussian operator. More information about these detectors can be found on [24].

2.1.2 Canny Edge Detector

The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intentions were to enhance the many edge detectors already out at the time he started his work. He was very successful in achieving his goal, and his ideas and methods can be found in his paper [16]. In it, he followed a list of criteria to improve current methods of edge detection. The first and most obvious is low error rate. It is important that edges occurring in images should not be missed and that there are no responses to non-edges. The second criterion is that the edge points should be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be minimum. A third criterion is to have only one response to a single edge. This was implemented because the first 2 were not substantial enough to completely eliminate the possibility of multiple responses to an edge. As said by John Canny:

> *1) Good detection. There should be a low probability of failing to mark real edge points, and low probability of falsely marking non-edge points. Since both these probabilities are monotonically decreasing functions of the output signal-to-noise ratio, this criterion corresponds to maximizing signal-to-noise ratio. 2) Good localization. The points marked as edge points by the operator should be as close as possible to the center of the true edge. 3) Only one response to a single edge. This is implicitly captured in the first criterion since when there are two responses to the same edge, one of them must be considered false. However, the mathematical form of the first criterion did not capture the multiple response requirement and it had to be made explicit.*

Based on these criteria, the Canny edge detector first smooths the image to eliminate any noise by a smoothing stage, typically Gaussian smoothing. It then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (non-maximum suppression). The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds and if the magnitude is below the lowest threshold, it is set to zero (made a non-edge). If the magnitude is above the higher threshold, it is made an edge. And if the magnitude is between the two thresholds, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above the higher threshold.

2.2 Color Segmentation

It was only a few centuries ago (between 1670 and 1672) when Isaac Newton first worked extensively on issues related to optics and the nature of light. Although Aristotle and other ancient scientists had already written on the nature of light and color vision, it was not until Newton that light was identified as the source of the color sensation. He demonstrated, clearly and with precision, that white light was formed by a band of colors (red, orange, yellow, green, blue, indigo and violet) that could be separated through a prism. This inspired the idea of presenting colors in terms of a mixing process that could reproduce them, and so was the origin of our modern color models (also called color order systems) that specify color mixtures with specific methods of color measurement to achieve human color vision modelling. A color

model describes colors as tuples of numbers, typically as three or four values or color components, e.g. RGB (Red Green Blue), HSV (Hue Saturation Value), CMYK (Cyan Magenta Yellow Key), HSL (Hue Saturation Lightness), YUV (Y - luma component (the brightness), U and V - chrominance (color) components).

In spite of the concepts and theory already discovered a long time ago, for many years all work done in digital image processing focused on segmenting gray scale images, due primarily to the fact that, until recently, computer systems were not powerful enough to display and manipulate large, full-color data sets. With the advent of more powerful and easily accessible hardware came a shift in the current research towards the more widely applicable and more complex problem of color segmentation. This is a common process in image analysis and furthermore computer vision, where the goal is to identify and isolate a specific color in an image. The final goal is to region split a whole or partial 2D image, making a clustering of the homogeneous color and texture regions with the objective of object recognition. This color segmentation process is nothing more nothing less than making use of the desired object physical properties such as light absorption, reflection, or emission spectra to identify the object on a image.

2.2.1 Color Model

To work with images and color representations, a color model is always used. To achieve the goal of color segmentation is preferable to use the HSV or HSL color model over alternative models such as RGB or CMYK, because of differences in the ways the models emulate how humans perceive color. RGB and CMYK are additive and subtractive models, respectively, modelling the way that primary color lights or pigments combine to form new colors when mixed. In the HSV model only one dimension of color is described: hue (**H**SV). Hue is described best as the words we normally think as describing color: red, purple, blue, etc. How light or dark a color is, is represented as lightness or value (HS**V**). In terms of a spectral definition of color, value describes the overall intensity or strength of the light. The last dimension of color that describes our response to color is saturation (H**S**V). Saturation refers to the dominance of hue in the color. The HSV model is commonly used in computer graphics applications, because a color can be chosen by first picking the hue, then selecting the desired saturation and value. A visualization method of the HSV model is the cone, see Fig. 2.3. In this representation, the hue is depicted as a three-dimensional conical formation of the color wheel. The saturation is represented by the distance from the center of a circular cross-section of the cone, and the value is the distance from the pointed end of the cone.

2.2.2 Environment Interferences

The color of an object depends on both the physics of the object in its environment and the characteristics of the perceiving entity. Physically, objects can be said to have the color of the light leaving their surfaces, which normally depends on the spectrum of the incident illumination and the reflectance properties of the surface, as well as potentially on the angles of illumination and viewing. In our perceiving entity (a typical low cost computer camera), this type of color interference is more accentuated than in a human eye since the human eye

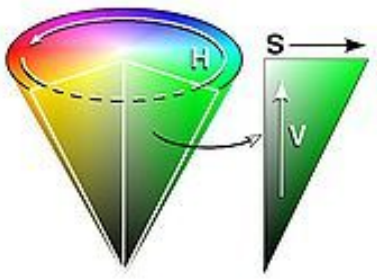


Figure 2.3: The conical representation of the HSV model is well-suited to visualizing the entire HSV color space as a single object.

is able to self-adapt to the environment. So in our case, the intensity and angle of the illumination can drastically change the perceived color of the same object, same as the viewing angle. Being the work centered around a mobile vehicle, this type of problems should always be considered. There are some objects that not only reflect light, but also transmit light or emit light themselves (e.g. traffic lights): this contributes to a better invariance of the color definition.

In order to achieve color segmentation the color model used in our image should be the HSV model. We isolate first in the hue dimension the range in which our desired color is, after we need to choose a range in saturation that better represents it and specify a minimum for the value component so that the final result is the most accurate, see Fig. 2.4.

Figure 2.4: Example of color segmentation. This is the result of a test performed with the values of $140 < H < 145$, $V > 0$ and $S > 50$. Image (a) is the original Lenna's image, (b) is the segmented image.

2.3 Template Matching

The detection and recognition of objects in images is a key research topic in the Computer Vision community. Within this area, the recognition and interpretation has attracted

increasing attention owing to the possibility of unveiling human perception mechanisms. One powerful tool that allow us to detect occurrences of a specific object in an image is Template Matching [13]. This digital image processing technique aims at finding small parts of an image which match a smaller template image. The matching provides a correlation, if the template image is a particular object image, the correlation result provides a way to estimate where that object can be in the image. Template is understood as anything fashioned, shaped, or designed to serve as a model, while matching is the comparison in respect of similarity or likeness.

The template matching method consists on using a convolution mask (which will be our template image), tailored to a specific feature of the search image, which we want to detect. This technique can be easily performed on grey images or edge images, or it can also be applied to full color images with the bearing of an heavy computational cost. The convolution output is highest at places where the image structure matches the mask structure.

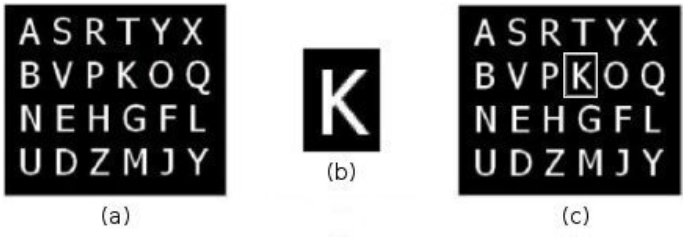


Figure 2.5: Template matching example: Applying a matching in the search image (a) with the template (b) gives a maximum response in the center pixel of the surrounding rectangle that marks the matched region in the result image (c).

Template Matching, as defined in [38], is conceptually a simple process. We need to match a template to an image, where the template is a sub-image that contains the shape we are trying to find. Accordingly, we center the template on a image point and make a comparison to see if the points in the template are a close match to those in the image. The procedure is repeated for the entire image and the point which led to the best match, maximum count, is deemed to be the point where the shape (given by the template) lies within the image. This process can be generalized to find, for example, templates of different size or orientation.

2.3.1 Matching methods

The method itself is normally implemented by first picking out a part of the search image to use as a template: We will call the search image $I(x, y)$, where (x, y) represent the coordinates of each pixel in the search image. We will call the template $T(x', y')$, where (x', y') represent the coordinates of each pixel in the template. We then simply move the center (or the origin) of the template $T(x', y')$ over each (x, y) point in the search image, and apply the method between the coefficients in $I(x, y)$ and $T(x', y')$ over the whole area spanned by the template. As all possible positions of the template with respect to the search image are considered, the position with the highest score is the best position. If the image is $W \times H$ and the template is $w \times h$ then the result must be $(W - w + 1) \times (H - h + 1)$. Some methods

described in [11] are done over template and the image patch: $x' = 0 \dots w-1$, $y' = 0 \dots h-1$. Being the result image $R(x, y)$:

Square difference matching method

The method matches the squared difference, so a perfect match will be 0 and bad matches will be large:

$$R_{sq_diff}(x, y) = \sum_{x',y'} [T(x', y') - I(x + x', y + y')]^2 \tag{2.4}$$

Correlation matching methods

The method multiplicative matches the template against the image, so a perfect match will be large and bad matches will be small or 0.

$$R_{ccorr}(x, y) = \sum_{x',y'} [T(x', y').I(x + x', y + y')]^2 \tag{2.5}$$

Correlation coefficient matching methods

The method matches a template relative to its mean against the image relative to its mean, so a perfect match will be 1 and a perfect mismatch will be -1; a value of 0 simply means that there is no correlation (random alignments).

$$R_{ccoef}(x, y) = \sum_{x',y'} [T'(x', y').I'(x + x', y + y')]^2 \tag{2.6}$$

$$T'(x', y') = T(x', y') - \frac{1}{(w.h) \sum_{x'',y''} T(x'', y'')} \tag{2.7}$$

$$I'(x + x', y + y') = I(x + x', y + y') - \frac{1}{(w.h) \sum_{x'',y''} I(x + x'', y + y'')} \tag{2.8}$$

Normalized methods

For each of the three methods just described, there are also normalized versions described by Rodgers [43]. The normalized methods are useful because they can help reduce the effects of lighting differences between the template and the image. In each case, the normalization coefficient is the same:

$$Z(x, y) = \sqrt{\sum_{x',y'} T(x', y')^2 . \sum_{x',y'} I(x + x', y + x')^2} \tag{2.9}$$

2.3.2 Template variability

A template image may also additionally exhibit some variability: not all of its instances are exactly equal, see Fig. 2.6. A simple example of template variability is related the corruption by additive noise. Another important example of variability is due to the different viewpoints from which a single object might be observed. Changes in illumination or sensor configuration may also cause significant variations. Another form of variability derives from intrinsic variability across physical object instances that causes variability of the corresponding image patterns: consider the many variations of faces, all of them sharing a basic structure, but also exhibiting marked differences. Another important source of variability can result from the temporal evolution of a single object, an interesting example being the mouth during speech. Many tasks of our everyday life require that we identify classes of objects in order to take appropriate actions despite of the significant variations that these objects may exhibit.

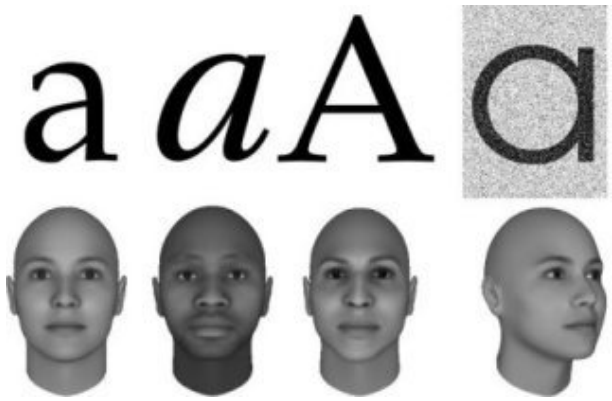


Figure 2.6: Template examples from two very common classes: characters and "characters", i.e. faces. Both classes exhibit intrinsic variability and can appear corrupted by noise, [11].

So it's clear that using only one template image may not be enough in a dynamic environment. The template matching technique itself isn't scale and rotation invariant, but with a good template images database, robust results can be achieved. For scale and rotation invariant object detection and recognition, there is a technique called SIFT [30] (Scale-Invariant Feature Transform). The SIFT features are local and based on the appearance of the object at particular interest points, the method is invariant to image scale and rotation and has some other benefits as well. In this work, SIFT is not described or implemented. It was a choice to use template matching instead of SIFT for two reasons: first - in the autonomous vehicle context, since the car is always horizontal, it isn't really needed a rotation invariant method, and the scale invariant property can be well achieved on template matching with different scaled templates; second - the time-consuming SIFT operations can be too heavy for the performance requirements on our present hardware.

2.4 Image Morphology

Morphological image processing (from the Greek word *morph*, meaning shape or form) has its foundation in the mathematically rigorous field of set theory. Based on mathematical

theory and techniques for the analysis and processing of geometrical structures, the techniques to expand its use to image processing was originally developed by Matheron and Serra [45] at the Ecole des Mines in Paris. Morphological transformations arise in a wide variety of contexts such as removing noise, isolating individual elements, and joining disparate elements in an image. Morphology can also be used to find intensity bumps or holes in an image and to find image gradients. There are significant works done by other researchers expanding the technique and fusing it with another concepts to achieve a significant complexity level capable of solving the most difficult tasks, [48] [40] [44]. In this thesis, i will only address the basic morphological operations in binary images and their use, keeping the theory inside the limits of our context.

The basic key process in morphological operations can be described as the local comparison of a kernel (i.e. a small window, a matrix NxM) with an image, where the results of this comparison will alter the pixels based on image's content. The kernel is called a structuring element (can be though as any pre-defined shape) and has an anchor point which normally is the center point. Thinking in a binary image which content define an object, when the image is probed with this small window i.e. structuring element, if at any given point the structural element is positioned and it shares some space with the object, then this will influence the result of the transformation. Generally speaking most morphological operations are based on simple expanding and shrinking pixel operations. The primary application of morphology occurs in binary images, though it's also used on grey level images. It can also be useful on range images, where grey levels represent the distance from the sensor to the objects in the scene, rather than the intensity of light reflected from them.

2.4.1 Morphology operators

The two basic morphological set transformations are erosion and dilation. These operators can also be used in parallel or/and series to produce other transformations including two very important ones that will be used in this work, called opening and closing. These transformations involve always the interaction between an image A (the object of interest) and a structuring set B (called the structuring element). We will now look into each one of these operations using some mathematical definitions described in [15]. At this moment we should recall that here we will only address binary morphological operations, the definitions should be extended to include gray-scale morphology.

Dilation

The dilation of A by B is denoted by $A \oplus B$ and is defined by:

$$A \oplus B = \bigcup_{b \in B} (A)_b \tag{2.10}$$

Erosion

The erosion of A by B is denoted by $A \ominus B$ and is defined as:

$$A \ominus B = \bigcup_{b \in B} (A)_{-b} \tag{2.11}$$

Opening

The opening of A by B is denoted by $A \circ B$ and is defined as:

$$A \circ B = (A \ominus B) \oplus B \tag{2.12}$$

Closing

The closing of A by B is denoted by $A \bullet B$ and is defined as:

$$A \bullet B = (A \oplus B) \ominus B \tag{2.13}$$

In practical terms, given an erosion of a binary image defined in (x, y), what happens beneath the hood is that the value of some point $p\,(x, y)$ is set to the minimum value of all of the points covered by the kernel when aligned at $p\,(x, y)$ (aligned at the anchor point). For the dilation operator, the equation is the same except that a maximum value is considered rather than the minimum.

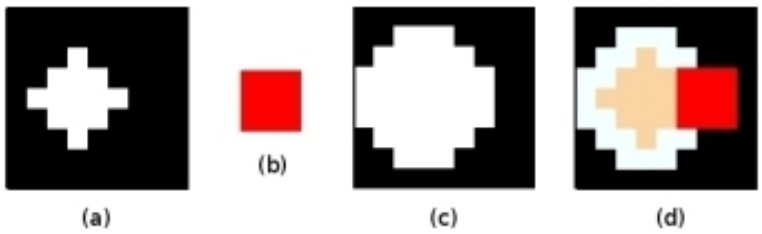


Figure 2.7: Defining the structuring element as the example (b) (the red square), the dilation of the image (a) by the structuring element in (b) results in the image (c). To perform the morphological dual operation, the erosion, we first pick the image (c), erode it by the structuring element (b), resulting the image (a). Image (d) puts (a),(b) and (c) together.

Chapter 3

Sensing Information, Fusion and Estimation

Robotics systems, [50], such as mobile platforms for planetary exploration, robotics arms in assembly lines, cars that travel autonomously on highways, actuated arms that assist surgeons, have in common that they are situated in the physical world, perceive their environments through sensors, and manipulate them. In opposite to manipulators in assembly lines that carry out the identical task day-in day-out, the most striking characteristic of the new robot systems is that they operate in increasingly unstructured environments that are inherently unpredictable. An assembly line is orders of magnitude more predictable and controllable than most of the environments of interest. Robotics is increasingly moving towards areas where sensor input becomes more and more important, and where the robot software has to be robust enough to cope with a range of situations often too many to anticipate them all. Uncertainty is a fact and arises from five different factors, as mentioned in [50]:

1. **Environments** - Physical worlds are inherently unpredictable. While the degree of uncertainty in well-structured environments such assembly lines is small, environments such as highways are highly dynamic and unpredictable.

2. **Sensors** - Sensors are inherently limited in what they can perceive. Limitations arise from two primary factors. First, range and resolution of a sensor is subject to physical laws. For example, the perceptual range of the spatial resolution of an camera's image is far too limited. Also, all cameras have inherent distortion especially the low cost ones, which are the ones used in our context. Second, sensors are subject to noise, which perturbs sensor measurements in unpredictable ways and hence limits the information that can be extracted from sensor measurements.

3. **Robots** - Robot actuation involves motors that are, at least to some extent, unpredictable, due to effects like control noise and deterioration resulting from ordinary use. Some actuators, such as heavy-duty industrial robot arms, are quite accurate. Others, like low-cost mobile robots, can be extremely inaccurate.

4. **Models** - Models are inherently inaccurate. Models are abstractions of the real world. As such, they only partially model the underlying physical processes of the robot and its environment. Model errors are a source of uncertainty that has largely been ignored

in robotics, despite of the fact that most robotic models used in the state of the art robotic systems are rather crude.

5. **Computation** - Robots are real-time systems, which limits the amount of computation that can be carried out. Many state of the art algorithms are approximations, achieving timely response through sacrificing accuracy.

In the field of robotics, due to uncertainty, it's common sense that a complete model of the environment is, in most cases of interest, unattainable.The perfect control of mechanical structures relative to this model is never a realistic assumption. There is also incoming noise from the sensors and actuators, all can lead to erroneous information or originate the lack of it. Sensing and perceiving the environment are means to compensate this unpredictable variations and also compensate the lack of the full model itself. Perception is the process of representing the sensory information in a task-oriented model of the world, [46], by processing data which is usually corrupted in various ways that complicate this process. In this chapter it is discussed the fusion and estimation of sensory information, as an introduction for the next chapter and early description of methods. These are essential aspects of a robotic system design.

The initial problem in sensory processing is data preprocessing and feature extraction, discussed in chapter 2. The role of these low-level processing tasks are to reduce noise, to remove any systematic errors and to enhance relevant aspects of the data. With fusion we combine this sensory data, in various dimensions, in order to achieve a better and more useful result than the one that would be achieved by using the data individually. This can be accomplished through various layers of increasing abstractions. Estimation is the interpretation of sensory data and integration of information across space and time. In some cases, the sensory information might also have to be temporally or spatially aligned for subsequent integration.

3.1 Linear Regression

A classic statistical problem is to determine the relationship between two variables, X and Y, where X is considered to be the independent variable and Y the dependent one. However, this nomenclature may vary according to the field of application, some of the names that the variables on the X and Y axis can have are: independent and dependent; predictor and predicted; carrier and response; input and output. Linear regression is a method that attempts to analyse this relationship and model it linearly, [35], commonly used in engineering problems as a tool for fitting a equation to a data set. The primary reasons why one needs to fit a regression equation to a data set are:

1. to describe the data;

2. to predict the response from the independent variable.

When the two variables data sets are plotted in a scatter-plot the simple linear regression here described is represented by a straight line, and this straight line is a good fit if the data

points are close to it. A linear regression line has an equation of the form:

$$Y = \alpha + \beta X \tag{3.1}$$

Above, X is the independent variable and Y is the dependent one. The slope of the line is β, and α is the intercept (the value of Y when $X = 0$). The most common method for fitting a regression line is the method of least squares. The general definition, [33], of the least squares problem is as follows:

Least Squares Problem Definition

Find $\mathbf{x}^*$, a local minimizer for

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}(f_i(\mathbf{x}))^2 ,$$

where $f_i : \mathbb{R}^n \mapsto \mathbb{R}$, $i = 1, \ldots, m$ are given functions, and $m \geq n$.

In the linear regression scope, this method calculates the best fitting model for the observed data by minimizing the sum of the squares of the vertical deviations, Fig. 3.1, from each data point to the line (if a point lies on the fitted line exactly, then its vertical deviation is 0), [52]. So, the best fit is the instance of the model for which the sum of squared residuals (residual is the vertical deviation) has its least value. This way, transforming X values only slides the data back and forth horizontally, and won't change the vertical distance between the data point and the curve. Transforming X values, therefore, won't change the best-fit values of the parameters, or their standard errors or confidence intervals.

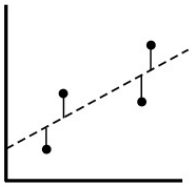


Figure 3.1: An illustration of the vertical deviations used in the method of least squares.

The deviations are first squared, then summed. The square is needed so there are no cancellations between positive and negative values, [29]. But not only, lets look at an example, lets say we have a possible line fit with two medium size deviations (say 5 units each), given by two points, and there is a different line fit with one small deviation (1 unit) and one large (9 units). A procedure that minimized the sum of the absolute value of the distances would have no preference over a line that was 5 units away from two points and one that was 1 unit away from one point and 9 units from another. The sum of the distances (more precisely, the sum of the absolute value of the distances) is 10 units in each case. A procedure that minimizes the sum of squares of the distances prefers to be 5 units away from two points (sum of squares = 50) rather than 1 unit away from one point and 9 units away from another (sum of squares = 82). The linear least squares fitting technique is the simplest and most commonly applied form of linear regression and provides a solution to the problem of finding

the best fitting straight line through a set of points. The least squares regression equation is the line for which the sum of squared residuals $E = (Y - Yi)^2$ is minimum. It is not necessary to perform trial and error attempts in order to get the best fitting line for the data set, as the line equation that minimizes the sum of squared residuals is algebraically deductible, [42]. With a data set of size n,

$$\overline{X} = \frac{\sum X_i}{n} \tag{3.2a}$$

$$\overline{Y} = \frac{\sum Y_i}{n} \tag{3.2b}$$

and

$$\alpha = \overline{Y} - \beta\,\overline{X} \tag{3.3}$$

We want to minimize:

$$\sum (Y_i \,-\, \beta\, X_i \,-\, \alpha)^2 \tag{3.4}$$

Using the coordinate transform $\alpha \,=\, \alpha_1 \,-\, \beta\,\overline{X}$, the resulting two independent quadratic expressions are:

$$n\,\alpha_1^2 \,-\, 2\,\alpha_1 \sum Y \tag{3.5}$$

$$\beta^2 \sum X^2 \,-\, \frac{(\sum X)^2}{n}\,\beta^2 - \,2\,\beta \sum (XY) \,+\, 2\,\frac{\sum X \sum Y}{n}\,\beta \tag{3.6}$$

The final resulting minimizing values are given by:

$$\alpha_1 \,=\, \frac{\sum Y}{n} \tag{3.7}$$

$$\beta \,=\, \frac{n \sum (XY) - \sum X \sum Y}{n \sum X^2 - (\sum X)^2} \tag{3.8}$$

From α_1, and β and the coordinate transform referred above, we get:

$$\alpha \,=\, \overline{Y} \,-\, \overline{X}\,\beta \tag{3.9}$$

Using the minimizing equations, a linear regression can now be applied to vectors X and Y, both with size n. First by determining α and β and then using the line equation $Y = \alpha + \beta X$ to predict the response from the independent variable.

3.2 Non-Linear Regression

In most cases of interest, a simple linear regression as described in the last section is not enough. This fitting technique is the simplest and most commonly applied form of linear regression to provide a solution to the problem of finding the best fitting straight line (but not only) through a set of points. But, in some scenarios we may need to model the data by an unknown curve or by a set of non-linear equations with parameters that can appear as functions. More specifically, in the context of the work supporting this thesis, the need to experiment and describe the data in various forms implied the inclusion of the non-linear regression in order to estimate any kind of relationship between the dependent (or response variable), and some list of independent variables. This section's final goal is to provide a method for data fitting, but allowing a pre-definition of any specific model to fit the data with (this data results, typically, from observations). The method's result is an estimation of the unknown parameters of the model. The intention here is not to describe the extensive iterative methods to solve non-linear regression problems, but instead give an oversight on the fundamentals, referring that the implementation done within this thesis support work is the Levenberg-Marquardt minimization adapted from Joachim Wuttke's Levenberg-Marquardt minimization and curve fitting implementation, [54]. The implementation is based on *lmdif* and other routines from the public-domain library *netlib::minpack*.

The Levenberg-Marquardt is one of the methods used to solve non-linear least squares problems, where the least-squares problem definition is often used to solve a set of non-linear equations. While the normal linear regression model may be written as equation 3.1, the more general non-linear regression model is as defined in [25]:

$$Y = \alpha + f(\beta, X) \tag{3.10}$$

where the function f, which relates the response to the predictors is not necessarily linear. The basis of the method is to approximate the model by a linear one and to refine the parameters by successive iterations, starting by choosing initial values for the parameters and then, refine the parameters iteratively, i.e. the values are obtained by successive approximations. There are many similarities to linear least squares, but also some significant differences. The goal of non-linear regression is again to adjust the values of the variables in the model to find a curve, that best predicts Y from X. Among the techniques used for unconstrained optimization of a non-linear objective function in the least squares sense, two of the most popular have been Gauss-Newton method and the already mentioned Levenberg-Marquardt. Extensive readings on methods for non-linear least square problems can be found on [33] and [12].

The method of Levenberg-Marquardt consists on an improved version of Gauss-Newton's method, who, in turn, is a variant of Newton's method. Basically, as described in [35], the Gauss-Newton method starts by computing how much the sum of squares changes when we make a small change in the value of each parameter. This represents the slope of the sum-of-squares surface at the point defined by the initial values. For a linear equation, this information is enough to determine the shape of the entire sum of squares surface, and thus calculate the best fit values in one step. With non-linear equations, the Gauss-Newton method won't find the best-fit values in one step, but that step usually improves the fit, [35]. After

repeating many iterations, the bottom is reached. The Levenberg-Marquardt algorithm is a modification of the Gauss-Newton algorithm and is a fairly widely used method for solving non-linear regression problems. While the Levenberg-Marquardt algorithm tends to converge somewhat slower than the Gauss Newton algorithm, it usually has fewer problems with divergence and is more robust.

The Gauss-Newton method is based on a linear approximation to the components of a vector function f (a linear model of f) in the neighbourhood of $\mathbf{x}$, as described in [33]. Provided that f has continuous second partial derivatives, for a small $\|h\|$, one can write its Taylor expansion,

$$f(\mathbf{x}+h) = f(\mathbf{x}) + J(\mathbf{x})h + O(\|h\|^2) \tag{3.11}$$

as

$$f(\mathbf{x}+h) \equiv f(\mathbf{x}) + J(\mathbf{x})h \tag{3.12}$$

where J is the Jacobian, which contains the first partial derivatives of the function components.

$$(J(\mathbf{x}))_{ij} = \frac{\partial f_i}{\partial x_j}(\mathbf{x}) \tag{3.13}$$

By using the above first-order approximation of f, we get the normal equations,

$$\left(J^\top J\right) h_{gn} = -J^\top f \tag{3.14}$$

where the solution is defined to be the Gauss-Newton increment, h_{gn}. Here, $J = J(\mathbf{x})$ and $f = f(\mathbf{x})$.

Levenberg (1944) and later Marquardt (1963) suggested to use a damped Gauss-Newton method. The Levenberg-Marquardt step h_{lm} is defined as the solution for:

$$\left(J^\top J + \mu I\right) h_{lm} = -J^\top f \tag{3.15}$$

with the damped parameter $\mu > 0$, being this a modification to 3.14.

There are several effects the damping parameter μ can have on the iterative process:

1. For all $\mu > 0$ the coefficient matrix is positive definite, and this ensures that h_{lm} is a descent direction.

2. For large values of μ we get

$$h_{lm} \simeq -\frac{1}{\mu} J^\top f \tag{3.16}$$

3. If μ is very small, then $h_{lm} \simeq h_{gn}$, which is a good step in the final stages of the iteration, when $\mathbf{x}$ is close to $\mathbf{x}^*$, the local minimizer for the least squares problem.

Being iterative, the Levenberg-Marquardt method first starts with an initial guess on the parameters (can be almost any value, typically 1). In each iteration the parameters are substituted by new estimates in order to reach the minimum residuals, in the least squares sense. In equation 3.15, I is the identity matrix. The (non-negative) damping factor, μ, is adjusted at each iteration. A smaller value brings the algorithm closer to the Gauss-Newton algorithm, whereas, increased, gives a step closer to the gradient descent direction.

Within the non-linear regression scope, one can describe previously a model and then use the iterative process to adjust the parameters of the model according to the data we are fitting, typically observation or measure vectors. The processes is based on the minimization of the residuals found between each iteration approximation and the supplied data. The method itself is robust and allows to achieve any type of regression that can be proved necessary.

3.3 Kalman Filter

Filtering has always been one of the most useful tools of engineering, since the real world is not perfect and noise is everywhere. Whenever the state of a specific system needs to be estimated from noisy sensor information, some kind of state estimator is normally employed to filter the data from the different sensors to produce an accurate estimate of the true system state. When the system dynamics and observation models are linear, this estimate may be achieved by the employment of the Kalman filter. This filter was first described in a paper published by Rudolf E. Kalman in 1960, [41].

The Kalman filter is an efficient recursive solution that allows to estimate the state of a dynamic linear system. The word recursive in the previous description means that, unlike certain data processing concepts, the Kalman filter does not require all previous data to be kept in storage and reprocessed every time a new measurement is taken. This is of vital importance to the practicality of the filter implementation. Each updated estimate of the state is computed from the previous estimate and the new input data, so only the previous estimate requires storage. In addition to eliminating the need for storing the entire past observed data, this makes Kalman filter computationally more efficient than computing the estimate directly from the entire past observed data at each step of the filtering process. The key in the Kalman filter is the correct state space formulation of the dynamic system. The fundamental concept is the notion of state, which can be though as some quantitative information (a set of numbers, a function, etc.) that is the least amount of data one has to know about the past behavior of the system in order to predict its future behavior. Thus, the filter is based on an underlying description of the dynamics in terms of state transition, i.e., one must specify how one state is transformed into another as time passes. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so, even when the precise nature of the modeled system is unknown.

In the state space model definition, x_k is the state vector at time k (the components of x_k are called state variables) and x_{k-1} represents the state past value (at time $k-1$). In a similar way, u_k is the control vector of the system at time k. In our notation, both of these vectors are vertical vectors, that is, they are of the form:

$$x_k = \begin{pmatrix} x_{1,k} \\ x_{2,k} \\ \vdots \\ x_{n,k} \end{pmatrix} \quad \text{and} \quad u_k = \begin{pmatrix} u_{1,k} \\ u_{2,k} \\ \vdots \\ u_{n,k} \end{pmatrix} \tag{3.17}$$

The Kalman model itself is defined by two stochastic equations: the process equation (3.18) and the measure equation (3.19), also known as the observation equation.

Process equation:

$$x_k = F_k x_{k-1} + B_k u_{k-1} + w_{k-1} \tag{3.18}$$

x_k - true state at time k u_{k-1} - input vector
F - state transition model w_{k-1} - process noise
B - control-input model

Observation equation:

$$z_k = H_k x_k + v_k \tag{3.19}$$

z_k - observation at time k v_k - observation noise
H - observation model

The filter model assumes that the true state at time k has evolved from the state at time $(k-1)$ according to equation (3.18). This process equation estimates actual state x_k, [53], by the sum of three mathematical plots. The first has a matrix F called state transition model, which relates the current state of the process x_k with the previous state x_{k-1}. The second plot has a matrix B called control-input model which relates the posterior state with the system induced control values (the inputs), represented by an input vector u_k. The last plot represents the process noise w_{k-1} which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_k,

$$p(w) \sim N(0, Q_k) \tag{3.20}$$

which is the process noise covariance matrix.

At time k an observation (or measurement) z_k of the true state x_k is made according to equation 3.19. The goal is to map the true state space into the observed space. This because, in the core of probabilistic robotics [50], state estimation addresses the problem of estimating

quantities from sensor data that are not directly observable, but that can be inferred. In most robotic applications, determining what to do is relatively easy if one only knew certain quantities. For example, moving a mobile robot is relatively easy if the exact location of the robot and all nearby obstacles are known. Unfortunately, these variables are not directly measurable. Instead, a robot has to rely on its sensors to gather this information. In the Kalman filter, the mapping of the true state space into the observed space is done with the observation model H_k that relates the true state x_k with the observation z_k, where v_k is the observation noise, which is assumed to be drawn from a zero mean multivariate normal distribution with covariance R_k:

$$p(v) \sim N(0, R_k) \tag{3.21}$$

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: predict equations and update equations. The predict equations are responsible for projecting forward (in time) the current state and the error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback, i.e., for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate. The predict equations can also be thought as time update equations, while the update equations can be thought of as measurement update equations, as mentioned in [53], where are also defined the equations used here.

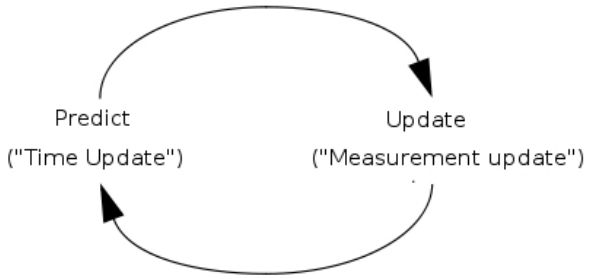


Figure 3.2: The Kalman filter cycle. The Predict projects the current state estimate ahead in time. The Update adjusts the projected estimate by an actual measurement at that time.

Predict equations:

$$\hat{x}_{k|pred} = F_k x_{k-1} + B_k u_{k-1} \tag{3.22}$$

$$P_{k|pred} = F_k P_{k-1} F_k^T + Q_{k-1} \tag{3.23}$$

$\hat{x}_{x|pred}$ - state estimate
$P_{x|pred}$ - error covariance matrix for the state estimate

In the update stage, the first task is to obtain the measure vector z_k, used to generate the measure residual $\tilde{y_k}$, via equation (3.24). The next step is to determine the measure residual covariance S_k, by equation (3.25). This is necessary to compute the Kalman gain, K_k, via equation (3.26), which is chosen to be the gain or blending factor that minimizes the *a posteriori* error covariance P_k. Finally, we can generate an *a posteriori* state estimate $\hat{x}_k$ by incorporating the measure residual $\tilde{y_k}$ and the Kalman gain K_k as in equation (3.27). The final step is to obtain an *a posteriori* error covariance estimate P_k via equation (3.28).

Update equations:

$$\tilde{y}_k = z_k - H_k\hat{x}_{k|pred} \tag{3.24}$$

$$S_k = H_k P_{k|pred} H_k^{\mathrm{T}} + R_k \tag{3.25}$$

$$K_k = P_{k|pred} H_k^{\mathrm{T}} S_k^{-1} \tag{3.26}$$

$$\hat{x}_k = \hat{x}_{k|pred} + K_k\tilde{y}_k \tag{3.27}$$

$$P_k = (I - K_k H_k) P_{k|pred} \tag{3.28}$$

$\tilde{y_k}$ - measure residual
S_k - measure residual covariance
z_k - measure vector
K_k - Kalman optimal gain
$\hat{x}_k$ - updated state estimate
P_k - updated estimate covariance

After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. The update equations can be arranged to only three equations and the Kalman cycle showed on Fig. 3.2 can now be completed with the Kalman equations, see Fig. 3.3. The initial state, and the noise vectors at each step are all assumed to be mutually independent.

Looking at the Kalman gain equation on Fig. 3.3, derived from equations (3.25) and (3.26), one can see that as the measurement error covariance R_k approaches zero, the gain K_k weights the residual more heavily, as in equation 3.29 (recall the state estimate $\hat{x}_k$ equation (3.27)). Indeed, if the errors are measure free, from equation (3.19) one has $x_k = {H_k}^{-1} z_k$ and so, no time update phase is needed at all. On the other hand, as the *a priori* estimate error covariance P_k approaches zero, the gain K_k weights the residual less heavily, as in equation (3.30). A higher Kalman gain means that the current measure z_k is more trusted to be correct while the predicted measurement $F_k\hat{x}_{k-1}$ is trusted less, this is opposite to a lower Kalman

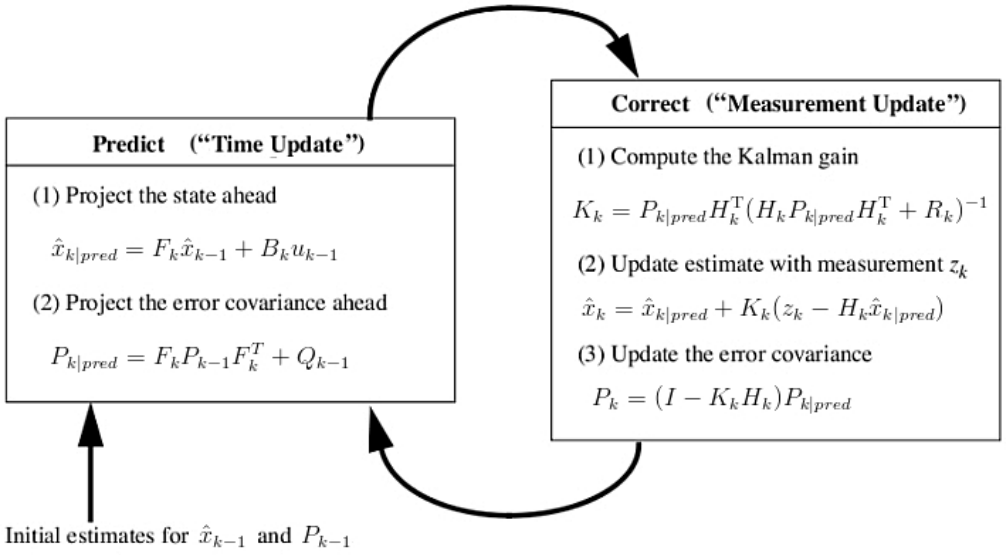


Figure 3.3: The Kalman filter complete operation, combining the high-level diagram of Fig. 3.2 with the Kalman equations.

gain that translates into a poor measure z_k and so less weighted in the final state estimate equation (3.27).

$$\lim_{R_k \to 0} K_k = H_k^{-1} \tag{3.29}$$

$$\lim_{P_k \to 0} K_k = 0 \tag{3.30}$$

The filter parameters and tuning should be introduced under certain considerations as referred by the authors of [53]. They mention that the measurement noise covariance R_k is usually measured prior to operation of the filter. Measuring the measurement error covariance R_k is generally practical (possible) because we need to be able to measure the process anyway (while operating the filter), so we should generally be able to take some off-line sample measurements in order to determine the variance of the measurement noise. On other hand, the determination of the process noise covariance Q_k is generally more difficult as we typically do not have the ability to directly observe the process we are estimating. Sometimes a relatively simple (poor) process model can produce acceptable results if one injects enough uncertainty into the process via the selection of Q_k. Certainly in this case one would hope that the process measurements are reliable. In either case, whether or not we have a rational basis for choosing the parameters, often times superior filter performance (statistically speaking) can be obtained by tuning the filter parameters Q_k and R_k . The tuning is usually performed off-line. Under conditions where Q_k and R_k are in fact constant, both the estimation error covariance P_k and the Kalman gain K_k will stabilize quickly and then remain constant. It is frequently the case however that the measurement error (in particular) does not remain constant. For example, when sighting other cars or entities in the camera system, the noise in measurements of nearby cars will be smaller than in far-away cars. Also, the process noise

Q is changed dynamically during filter operation becoming Q_k in order to adjust to different dynamics. For example, in the case of tracking the head of a user of a 3D virtual environment we might reduce the magnitude of Q_k if the user seems to be moving slowly, and increase the magnitude if the dynamics start changing rapidly. In such cases Q_k might be chosen to account for both uncertainty about the users intentions and uncertainty in the model.

3.4 Extended - Kalman Filter

In the extended Kalman filter, the state transition and observation models need not be linear functions of the state but may instead be differentiable functions, and it's used when one attempts to model non-linear systems. A function f can be used to compute the predicted state from the previous estimate and similarly, a function h can be used to compute the predicted measurement from the predicted state. However, f and h cannot be applied to the covariance directly. Instead a matrix of partial derivatives (the Jacobian) is computed. So that at each time-step the Jacobian is evaluated with current predicted states, tending to linearize the non-linear function around the current estimate.

Process equation:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \tag{3.31}$$

x_k - true state at time k w_{k-1} - process noise
f - state transition model
u_{k-1} - input vector

Observation equation:

$$z_k = h(x_k) + v_k \tag{3.32}$$

z_k - observation at time k v_k - observation noise
h - observation model

Where wk and vk are the process and observation noises which are both assumed to be zero mean multivariate Gaussian noises with covariance Qk and Rk respectively.

Predict equations:

$$\hat{x}_{k|pred} = f(x_{k-1}, u_{k-1}) \tag{3.33}$$

$$P_{k|pred} = F_k P_{k-1} F_k^T + Q_{k-1} \tag{3.34}$$

$\hat{x}_{x|pred}$ - state estimate
$P_{x|pred}$ - error covariance estimate

Update equations:

$$\tilde{y}_k = z_k - h(\hat{x}_{k|pred}) \tag{3.35}$$

$$S_k = H_k P_{k|pred} H_k^T + R_k \tag{3.36}$$

$$K_k = P_{k|pred} H_k^T S_k^{-1} \tag{3.37}$$

$$\hat{x}_k = \hat{x}_{k|pred} + K_k \tilde{y}_k \tag{3.38}$$

$$P_k = (I - K_k H_k) P_{k|pred} \tag{3.39}$$

$\tilde{y_k}$ - measure residual	K_k - Kalman optimal gain
S_k - measure residual covariance	$\hat{x}_k$ - updated state estimate
z_k - measure vector	P_k - updated estimate covariance

In the extended Kalman filter, the state transition and observation matrices, that are to be applied to the covariance matrices, are defined to be the following Jacobians:

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1|k-1}, u_k} \tag{3.40}$$

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k|k-1}} \tag{3.41}$$

Chapter 4

Visual Perception System

A person's knowledge of the surrounding world is mediated through senses such as vision, smell, taste, touch, etc. In a visual perception process, light emanating from observed objects in the environment is used to infer a wealth of information about it. We are able to "see" depth and shapes of objects, to segment a scene into distinct objects or to perceive the mood of a partner. All of this, is based on images, i.e., two-dimensional distributions of intensities, which by themselves don't contain any information about depth, shape or moods. And even using a single eye we can realize all of them. The fact that we are able to see this is one of the humans brain most fascinating abilities.

The visual perception process itself cannot be accomplished without environment assumptions, i.e., the concept itself implies abilities that use internal models or representations of the environment. It is not possible to extract any knowledge from an image without a representation of what we are looking for. Furthermore, that is never done in the absence of noise and ambiguity. It is considered that perception is the task-oriented interpretation of data and integration of sensor information. Note that by sensing information we address information fusion, estimation, as a way to achieve a further step to perception. The input to a perception process is typically obtained from: digital data; a partial model of the environment (a world model), that includes information about the state of the robot and other relevant entities in the external world, [46].

The perception process can be a vague concept without an stimulus, i.e., its final result can never be an omniscient state of knowledge about its environment, only a specific goal oriented representation of it. The process provides information about the environment state and the robot state, as basis for control, decision making and interaction. The quality of the process results are of the most influence on the robot overall abilities. The limitations on the artificial intelligence of a robot agent, are directly connected to the quality and versatility of their perceptual systems. This is particularly true for the cases where the environment is only partially observed, extremely noisy and ambiguous.

In this chapter the Computer Vision algorithms implemented in ROTA are presented. Initially it is described the extraction of the road geometry from monocular images by the lane detection system, followed by the traffic lights, road maintenance area and obstacle detection. Then it is described the Inverse Perspective Mapping method, which allows that all

the detected elements can be mapped into real world coordinates. Overall, the work done at the vision perception level is intended to innovate and provide more accurate sensorial abilities, paving the way to more complex high level decision and planning.

4.1 Lane Detection System

Given the performance requirements in our context, it was decided that the road detection was to be implemented through the execution of two medium level tasks: 1) Global search of the lane markings 2) Local search of the lane markings. The local search can be thought of as an effort to maintain a state about the previously detected boundaries. This allows to reduce the system computational costs, by minimizing the region where to apply the detection algorithms. This is accomplished with the guidance of the region of interest for low-level feature detection algorithms. So, in our context, each lane marking is an entity that has its own characteristics and a region of interest associated. Since the detection of the lane markings is based on distinct transitions of the image intensities, a well defined road boundary is also considered a lane marking and so a road delimiter.

4.1.1 Lane Markings Detection and Tracking

In order to detect the lane markings of the observed lane in the image captured by the camera the Canny edge detector is firstly used. So, no noise reduction implementation is necessary since it's already included in the first stage of the algorithm. Also, the use of the Canny algorithm allows to identify high intensities transitions (edges) with elimination of false-positives and multiple responses to a single edge, [16], resulting in overall detection robustness. Since colors are not considered, the resulting approach to the lane markings detection is also applicable to markings and roads with different colors. The result of the Canny algorithm, see Fig. 4.1, is a binary image with its pixels values set to "one" if they are edges and "zero" if they are not.

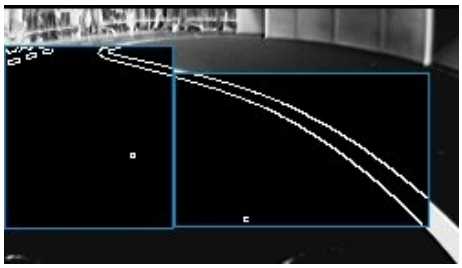


Figure 4.1: The Canny edge detector is applied to both regions of interest delimited with a thin light blue line. This oriented approach consists on one dynamic region for each lane marking to be detected. At each frame's step, the lane markings regions of interest are readjusted.

After the low-level pre-processing, the image is searched for continuous or discontinuous (dashed) lines represented by aggregations of edges points. The discontinuous lane markings

detection feature represents a challenge, and was one of the most consuming tasks, in terms of design and implementation. The fact is that the marks of a discontinuous lane marking are not connected, and the distance between these marks is, in some cases, equal or greater to the distance to any other road artifact [1]. With the robot's movement, the discontinuous marks and the spaces between them, "move downwards" in the image (in the case of a forward moving robot). Thus, a point previously detected as a discontinuous lane marking point, will eventually lay down on the interval between the marks, difficulting the process. The dashed line that represents a discontinuous lane marking is here considered as a full line with edge points and non-edge ones.

4.1.2 Global Search for Lane Markings

The global search algorithm starts by picking one scanline, described by a starting point $s(x, y)$, and a direction where to conduct the search. This choice will be better described further ahead in this section. Horizontal search directions are used to search for our candidate edge points. This directional searches with initial point, direction and length are often called scanlines. Their result, in this context, translates into the occurrence of edge locations. When the probing of an scanline ends without returning any edge occurrences, a smaller search is done in its perpendicular direction, by considering a small neighborhood. This results in a search not limited exclusively by the horizontal direction of the described scanlines. In the current detection architecture, for each frame's step, the global search is done only for one lane marking at a time. The global search result is a set of data points, that are to be classified later.

When a global search is done on the image, the probe of the first scanline can result in an occurrence of an edge location or the absence of one. Either case, the global search algorithm continues the search alternately above the last scanline or bellow the penultimate one, see Fig. 4.2. This is performed until the global search reaches a predefined maximum or minimum. The vertical resolution of the search is defined as the parameter h. Considering that y_i is the ordinate of the initial point of the iˆth scanline, one have:

$$y_i = \begin{cases} y_1 + \frac{n-1}{2}h & \text{i odd} \\ y_1 - \frac{n}{2}h & \text{i even} \end{cases}$$

The second scanline has the same length as the first, so that a localized noisy pixel that could affect the whole search, can be discarded at this step. This is used essentially to validate the search, so that if they are too distant from each other, the point that has the greater distance to the respective scanline initial point, is discarded.

For each scanline used in the search, if an edge is encountered, then the point is added to the resulting search candidate points. During the search, if three of them are found, a linear regression is applied to estimate where the next scanline initial point should be positioned. For each point added to the candidate list, the regression is modified to include the new one

[1] Note that the image isn't without perspective, so objects appear smaller as their distance from the observer increases and also the size of an object's dimensions along the line of sight are relatively shorter than dimensions directly across the line of sight. To agravate this the image provided by ROTA's cameras has a mixture of sphere and fish eye distortion; Here the distance considered is in the euclidean space.

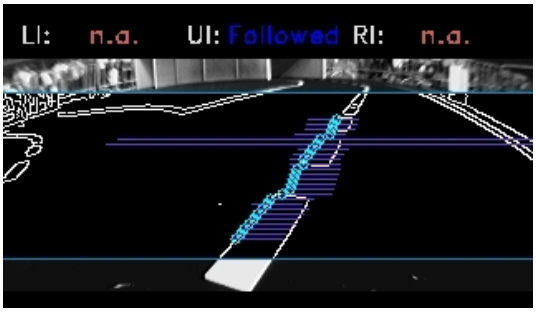


Figure 4.2: Here, with the Canny edge detector applied to the region of interest delimited with a thin light blue line, the search process is illustrated with the original scanlines length and position painted in purple. The direction of the search is from the left to the right and the initial point is on the left middle of the image. The scan lines positions are decided as the algorithm progresses.

and a small compensator is changed, to produce better results. If an edge point is not found, nothing is added to the regression analysis, although, the correspondent point resulting from the regression is kept for further analysis. As the algorithm progresses, the scanlines are placed based on an estimation, of how the searched line is in the image. This placement results in a search method that works better for initial search points positioned in the vertical middle of the lane marking's region of interest.

Assisted Global Search for Lane Markings

A global search can also include knowledge of the already detected lane markings and use them as parameters for the search. The insertion of auxiliary lane markings for the global search of new ones result in a selection of the initial points and length for the scanlines that are to be used, thereby increasing search speed. Having the initial positions and lengths for the scanlines, the region of the search is completely defined. So by using the knowledge of the already known lane markings, the algorithm knows exactly where to look. The result is a more refined search.

Parameters Selection

When the global search is conducted without the assistance of already known markings, an initial search point centered in the middle of the possible road image area is used, e.g. when the system is initially turned on. The direction of the search (left or right) is then alternated, until a new road element is found. This is achieved by the use of a priority queue for the global detections. For example, in the case where only the discontinuous lane marking is detected but none of the right or left continuous markings are, each invocation to the algorithm alternates between a right side search for new markings and a left side search. For the cases where the robot does not have any knowledge about where the road is, an initial search point, centered in the middle of the possible road image area, is used.

For the overall method efficiency of the global searches, the y component of the initial

search point $s(x, y)$, should be somewhere in the middle of where currently is expected to find any lane markings in the image. For the general case, the vertical middle point of the region of interest can be assigned to y. However, for the cases where an auxiliary lane marking is partially observed in height and considered for the assistance of a search, a lower or higher y is assigned, since the y component of the initial search point is centered on the middle of the auxiliary lane marking. The parameters of the global search, such as initial search point, direction and vertical dimension of the search area, are changed dynamically with the feedback of the whole road detection system.

The key point in this global search method is that although the search is intrinsically oriented by the lane markings more required at the moment, by using an internal model, the found lane marking isn't necessarily the same has the searched one. Instead, the new lane marking is classified for model fitting. The classification is made in terms of continuity or discontinuity, in the second case the number of discontinuities also is considered, as for the total number of edge occurrences.

4.1.3 Local Search for Lane Markings

The lane markings local search algorithm is a method that performs local scanning around a small neighbourhood of the previously detected lane marking points. This is made in order to maintain and improve their detection in the image. The method can consider only the first points found with the scanlines without affecting its reliability, reducing the cost of the search in computational terms.

For each previously detected marking position in the image, a small neighborhood is scanned for each of its points, in order to maintain the previous state of the detection. If a point is found then it's updated in the correspondent lane marking structure. If, on the other hand, it isn't found, the final result will include a posteriori estimate by regression analysis. In the case of extrapolation, a linear regression is applied to the nearest points with a small compensator dependent on the vertical axis, in order to best adapt the detection to the radial distortion of the cameras used and possible curvature of the marking.

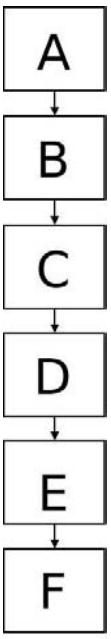
Adapting to the environment

In the local search, the number of the searched points of the markings are increased or decreased to improve the quality of their detection. The dimension of the search area changes dynamically, so that if the current detection conducted by the algorithm results in a positive, but low quality detection, i.e., there should be more space where to detect the markings, a regression analysis is used to estimate where new points can be searched. In the other hand, if the points that are not found are only the ones on the lane marking's extremities, then the local search considers that the same marking was found, with its length diminished. In practice the algorithm increases, decreases or maintains the number of scanlines above and bellow the already detected marking in the image, for each cycle. The number of the scanlines increment is a given step parameter. Good results are obtained if set to 1. This improvement at each cycle is efficient and useful, since there are cases where a very low quality detection caught by the global search algorithm is drastically improved over time, becoming a high

quality detection. It also proved effective, when the lane marking becomes partially occluded, since the size of the detection area increases/decreases gradually.

4.1.4 Architecture

The system is implemented with the following architecture flow:



A - Parameters selection for the upcoming search, selection is made upon a desired lane marking.

B - Merging of the Regions of Interest for low-level processing.

C - Appliance of the Canny edge detector on previously determined regions of interest .

D - Global Search for a lane marking representation in the image.

E - Local Search for all currently detected lane markings.

F - Validation of the lane markings detection by use of the internal lane model

In stage A, regions of interest and the parameters for one intuitive lane marking global search are selected, also including a selection criteria based the on already followed lane markings in order to assist this upcoming search. The parameters are given by a particular lane marking search, one that is not followed and is the first on the priority queue. Note that this needs to be done first, before stage B, so that the resulting region of interest selected, is available to stage B. Later, in stage D, the global search is made within the parameters decided in A. After a search with positive results, where a lane marking was encountered, the classification takes place. This classification is done inside stage D, and confers an independency between the desired world missing lane and the final search results.

Stage B is dedicated to the organization of the list of ROI's that are to be passed to the edge detection algorithm, in order to reduce the time cost of stage C and avoid that the same image is processed twice on the same place. The stage B collects all regions of interest and checks if any two of them intersect. If so, they are merged as a new region of interest and the process continues, until there are no more region intersections on the list. Stage C is where the Canny edge detector is applied, and stage E is where the local adaptive searches are made in order to maintain or improve the current lane markings detections. In stage F, the lane markings are compared and validated against a model, so that the system can correct itself. The model is very simple, in the way that it only describes the basic features of a typical road lane. The model says that the lane markings can be only in the order: continuous-discontinuous-continuous, or continuous-continuous. In the inverse perspective mapping this

validation task can be improved with real distances.

It is worth mentioning that the choice to limit the system to one global search for an lane marking, for each cycle, is driven by the fact that the image processing becomes quite computationally heavy and could origin frame losses. It is more valuable to retrieve a new frame acquisition between each global search. This also introduces some extra confidence on the system, since the lane markings followed, that are used each time a global search is made, result of derived past knowledge, and are also confirmed by the current frame detection.

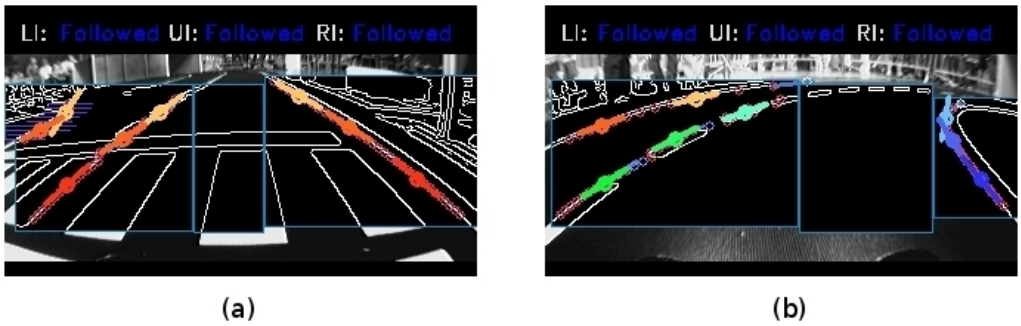


Figure 4.3: The images above were taken during the execution of the lane detection algorithm. In image (a) can be seen a successful global search driven towards locating the left continuous lane marking. In the figure, a straight section in image (a) and a curved section in image (b).

Inside the Tunnel

The lane detection system is also successful in detecting the boundaries that lay inside the tunnel, making it easy to navigate inside it. This because there is a clear high intensity transition in the image delimiting the tunnel walls and the lane. This is seen as continuous lines and therefore considered as normal boundaries by the Vision Perception System.

4.1.5 Curvature Estimation

The lane detection system naturally converges to a solution through a feedback loop control, since it corrects itself through the selection of parameters for search, classification and model validation. What happens is that the resulting lane markings, that are the output of the system, are used to infer what is about to be searched in the next cycle. Then the independent classification and model validation result in a dynamic system. When the detection system converges to a solution, the result is a set of point measures for each one of the lane markings found by the detection system. They only provide the limits of the lane ahead in the detection resolution. So, in order to achieve higher levels of environment perception there is the need to do data regression analysis. The first thing one can wish to determine about the lane ahead of the vehicle is the type of section and curvature. This is not an easy task as it may seem at first glance.

There is a main issue to be considered: there can be three or less observed lane markings in the image and there has to be a way of knowing which one is more accurate in representing the lane characteristics. Thus, the first step is to do regression analysis in order to fit the data to some model that can provide useful information about the lane markings. Different methods were tested during the experimental research phase. The first idea was to approximate each lane marking by three distinct regions and in each region consider a linear regression applied to its data points, see Fig. 4.3. However, the number of regions increased the erroneous information extracted by each region points, since their number decreases as for the regions increase, and the variance of each marking's region (size, position) brought more complexity to the analysis process. The linear regressions information, didn't proved useful or the best method. The next step was to use data regression analysis to fit to the detected points to equations of third degree. The solution was to approximate each lane marking to a polynomial, where A, B, C, D are constants, and x is a variable:

$$Ax^3 + Bx^2 + Cx + D \tag{4.1}$$

Inherently to the lane markings on the track, or their representation on the image, are in some cases a curve and a line, or in another cases a S-section that is composed of a curve-line-curve (with curves in different directions). This implied that the approximation of a second order polynomial function of the type $Ax^2 + Bx + c$ was not enough, since this interpolation can't deal with those cases sections. In Fig. 4.3, image (b) the car is already performing a curve but the right lane marking isn't a perfect curve, since a straight line section can be seen in the lower part. In equation 4.1, if $A \neq 0$ the model allows the lane marking to have two different curvatures and a transition, and if $A = 0$ there is only one curvature associated. The analysis of the function tells that if $A = 0$ and $B = 0$, the fitted lane marking is approximated by a linear equation $Cx + D$, and therefore, best fitted as a straight line. Test results showed that to determine a curvature, the parameter B can be taken in account with good results. This parameter is directly related to the polynomial curvature degree. So, the method is such that if B is above a defined threshold $thres$, the corresponding marking lane is considered to be a positive right curve, and if bellow $-thres$ is considered to be a left curve. If B is inside this interval, this means that the lane marking is related to a straight section. Experimental results show that the proposed method is valid. The regression analysis is performed using the method of Levenberg-Marquardt, [54].

4.1.6 Error Estimation

The error of a lane marking should be associated with: 1) the fitting variance, 2) the number of points detected. The second is of extreme importance, since a low number of points will always result in small variances to any given lane marking. The method that is used to estimate instant error of a detected lane marking uses the following equation:

$$LError = |\sigma^2 + (Max_{pts} - Cur_{pts}) * K| \tag{4.2}$$

where Max_{pts} is the maximum number of points that can be detected, Cur_{pts} are the current detected points and K is a constant used to increase the influence of the number of detected points in the error estimate. K should have a bigger value than 5 but if too large, the error

estimate will ignore the regression fitting variance. This variance is represented by σ^2, and is given by:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x}_i)^2 \tag{4.3}$$

where the data points detected are used as x_i and $\overline{x}_i$ are the correspondent estimated points in the regression analysis. N is the number of current points detected.

The information about the whole road lane has to be at all times derived from the information of each lane marking detected. Since each one has an error associated, the whole lane final representation is weighted between them, based on their instant error. So for each lane marking j, the weight confidence $LConf_j$ is given by:

$$LConf_j = 1 - \frac{LError_j}{\sum_{i=1}^{M}(LError_i)} \tag{4.4}$$

where $\sum_{i=1}^{M}(LError_i)$ is the sum of all lane markings instant errors, and M is the number of visible markings. This value is more useful when normalized, and so, by determining the sum of the visible lane markings confidence as $\sum_{i=1}^{M}(LConf_i)$, the normalization $NLConf_j$ between 0 and 1 is achieved by:

$$NLConf_j = \frac{LConf_j}{\sum_{i=1}^{M}(LConf_i)} \tag{4.5}$$

This supplies a way to determine how each lane marking detection can contribute to the perception system. The contributions are already normalized between 0 and 1, and the sum of these normalizations is always 1. The instantaneous information resulting from the analysis of the detected lane markings is considered also to have am error associated, which is the average, not weighted, of the lane markings errors. This provides more information about the detection system, opposite to the use of only the best individual lane marking detection.

The lane section's curvature analysis is done at each cycle recursively, based on the Kalman filter approach, the instantaneous contributions of each lane marking and the error estimation are considered. The lane section analysis aims at providing the current section type. The associated error reflects the quality of the detection itself and the confidence on the interpretation of the current lane. The analysis of the B variable of equation 4.1, is done for the whole detected lane, this will be B_L. It takes into consideration each lane marking B_j, according to the following equation:

$$B_L = \sum_{j=1}^{M}(NLConf_j * B_j) \tag{4.6}$$

The error associated to B_L is represented by S_L, and is an average of the detected lane markings errors, obtained from the various $LError_j$. If three lane markings are detected then this error should be smaller then the one obtained when only one lane marking is detected. To reflect this, the S_L error will be $S_L = S_L * D$, with $D = 1$ if all three lane markings are currently detected, $D = 2$ if only two are detected and $D = 3$ if only one is detected. The values of B_L and S_L are analysed each cycle. They will be used as measures to update the

variables $LaneB$ and $LaneSE$ in a Kalman filter approach, where $LaneB$ results from the filtering and smoothing of B_L and $LaneSE$ from the filtering and smoothing of S_L.

The lane section is assumed to not change drastically in time, so that the future state of the section is at most times equal to the one detected before. And when it changes, it does it smoothly: if the car is on a left curved section, then, before the lane changes to a right section, there should be a straight segment that transits between those two. This is supported by constrains from the real world, for the construction of lanes. The transition model here is such that $LaneB_k = LaneB_{k-1}$, considering that discreet time is given by k, as in the Kalman state approach. The process static error P_e is used to ensure that past estimations have larger errors associated. For each cycle, the $LaneSE$ error advances in time by taking this into account:

$$LaneSE_k = LaneSE_{k-1} + P_e \tag{4.7}$$

The Kalman gain K that will balance the measure and the estimation is determined as follows:

$$K = \frac{LaneSE_k}{LaneSE_k + S_L} \tag{4.8}$$

The final value of $LaneB$ is updated using the current measure B_L.

$$LaneB_k = LaneB_{k-1} + K * (B_L - LaneB_{k-1}) \tag{4.9}$$

This is succeeded by the update on the estimated error $LaneSE_k$:

$$LaneSE_k = (1 - K) * LaneSE_k \tag{4.10}$$

If no lane is detected, and so the robot is momentarily "blind", the $LaneSE$ error is increased, by adding a larger process error BLP_e that reflects the loss of confidence that the detection system has on the current interpreted section. Accordingly, the equations from 4.7 to 4.10, are replaced by:

$$LaneSE_k = LaneSE_{k-1} + BLP_e \tag{4.11}$$

Using this method, the confidence on the current lane analysis is represented by $LaneSE$. An experimental test analysis is described next, see Fig. 4.4. This is a difficult path for the robot, so it deals with difficult scenarios for the Lane Detection System. In image (a), from (1) to (2), the robot camera can't momentarily "see" any of the three lane markings, so, as it's "blind", the error $LaneSE$ keeps increasing, meaning that: a) the current belief keeps getting more and more poor, as time increases b) when a new measure is taken, despite of its error, its value will be taken heavily in account, given its current poor belief on the state of the world, see Fig. 4.5. This measure weight will return to normal, as time passes. An important aspect is that the lane markings having the worst detection results have in fact, less importance, as seen in Fig. 4.6. The detection of the discontinuous marking has always less points than a comparative continuous line and the farthest a lane marking is, the smaller the region of image it occupies and therefore the marking will have a smaller importance for the process. In this experiment there is an exception to this, in the tunnel area, where momentarily the use of the robot front lights results in a bad detection of the right marking,

the closest one. This is because the intensity of the illumination in the near tunnel walls, makes the edge detection fail for the right marking and therefore the discontinuous marking is taken in more account, as can be seen on Fig. 4.5 and in Fig. 4.6.

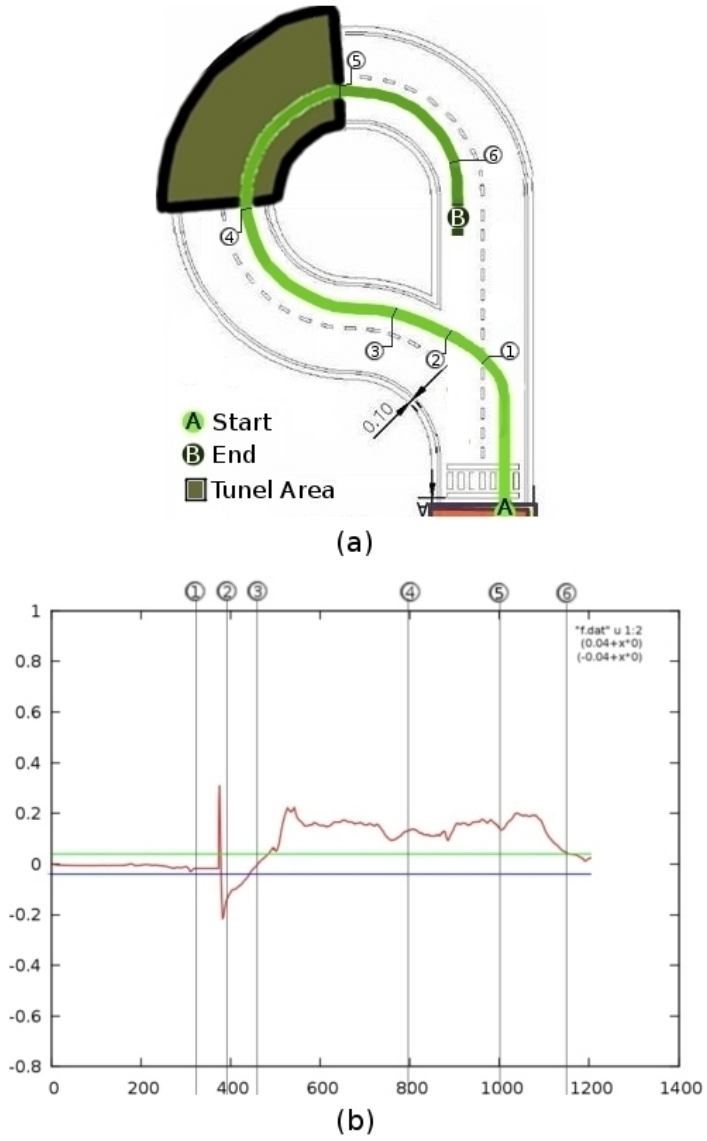


Figure 4.4: Above is represented, in image (a), the traveled path used for the experimental analysis. The robot starts moving at cycle 150. Between (1) and (2) the robot can't detect any lane marking. The estimated parameter $LaneB$ is showed in the scatterplot (b). The upper horizontal line is the minimum threshold to a right curve, the lower one is the maximum threshold to a left one. If between both, the section is interpreted as a straight one.

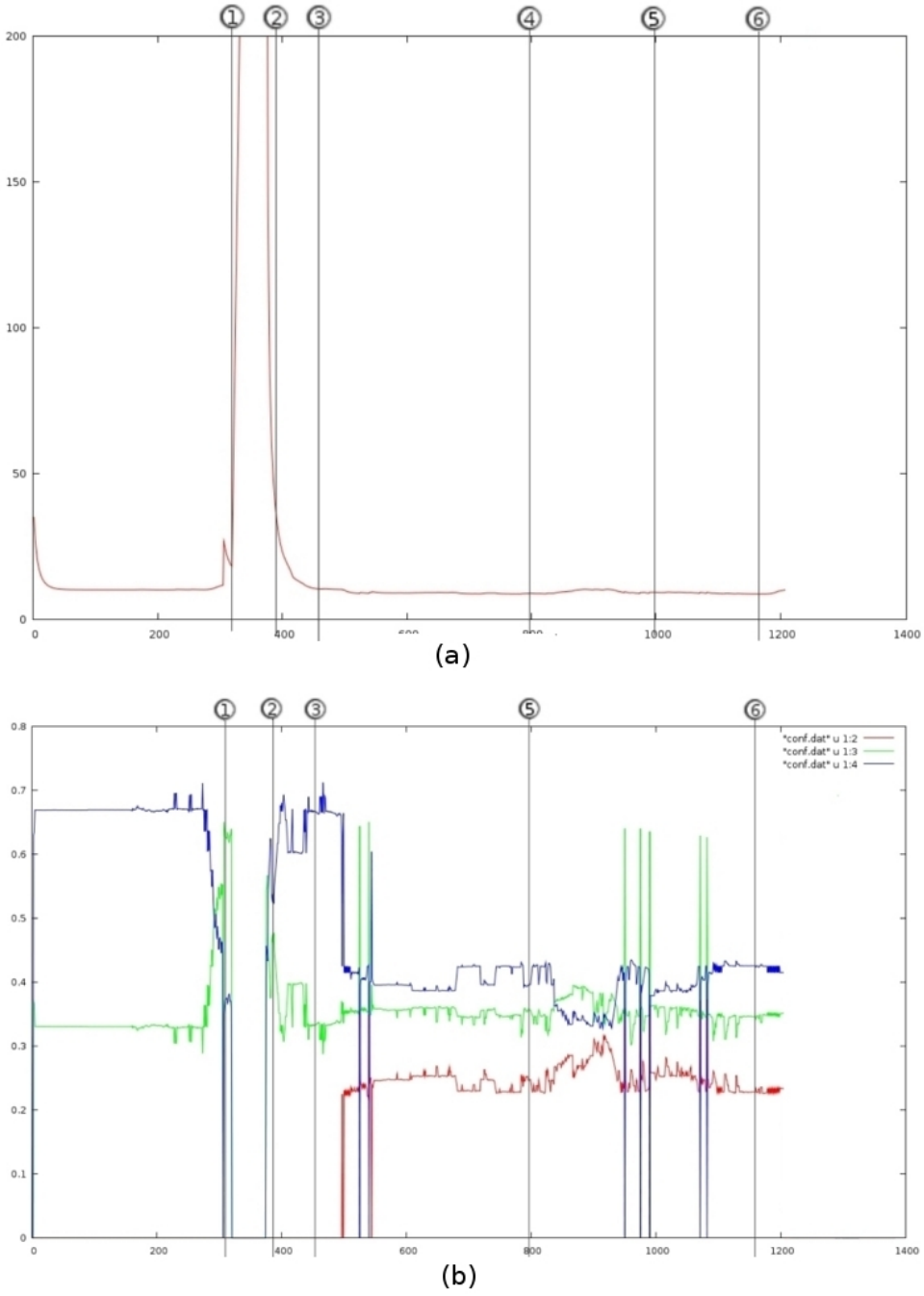


Figure 4.5: Image (a) contains the scatterplot for the estimation of $LaneSE$. Between (1) and (2) the robot is blind and so the error increases drastically. In image (b), are shown the normalized confidences of each lane marking, $NLConf_{lm}$. The blue line is the continuous right, the red is the left and the green is the middle marking. These scatterplots relate to the experimental path in Fig. 4.4.

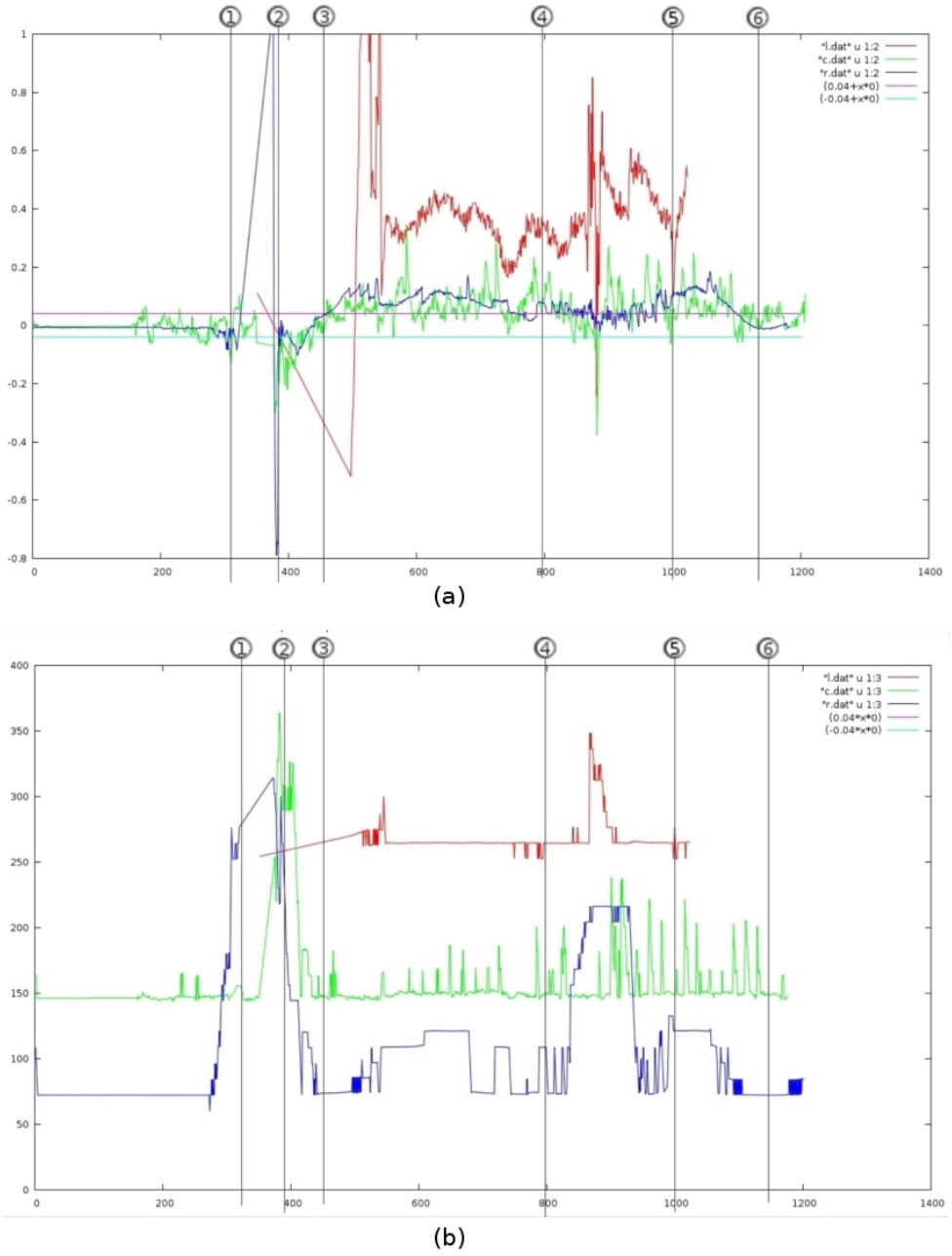


Figure 4.6: The measures of B for each lane marking are in image (a). The errors associated are in image (b). The large discrepancies that can be found, for example for the blue line between (1) and (2), are related to poor detections of lane markings and therefore have large error associated. The measure is discarded for these cases. Also note that the points are discreet in time, and in the scatterplots they are connected by a continuous line for a best view. The blue line stands for the continuous right, the red is the left and the green is the middle marking. These scatterplots relate to the experimental path in Fig. 4.4. Between (1) and (2) the robot can't detect any lane marking. Starts by loosing the right marking first and then the discontinuous. This can be seen properly in image (b). In image (a), the upper horizontal line is the threshold to a right curve, the lower is the threshold to a left one. If between both, it is interpreted a straight section.

4.2 Traffic Lights Detection

In the previous implementation of the traffic lights detection algorithm in ROTA color segmentation was used, with a posterior analysis of possible vertical and horizontal symmetries centered on the center of mass of the color blob that resulted from the segmentation. Still, the decisive element of the search was based on color, as for all previous implementations of detection algorithms in the ROTA scope. A count was done for the total number of pixels in the image, that had their color associated with the ones that the signals were assumed to have. The signals associated with the greater count, would be the selected one.

After research on the subject, it was accepted that the use of an extension to the template matching method should be the most efficient method for the traffic signals detection, given our experimental platform. The template matching is performed using the normalized correlation matching method, [11]. Experimental tests showed that this method is the one that achieves the best results, within this context.

Figure 4.7: Above are displayed four group types of template signs that are used for the example in Fig. 4.8. The different size and low quality of the images templates are intentional.

The implemented method for the traffic lights detection is based on the matching of each image template (here the traffic signals images), see Fig. 4.7, with the image frame acquired from the upper ROTA camera. The result of each matching is a normalized value between 0 and 1 that relates to how much confidence the matching algorithm has on the presence of the corresponding traffic signal in the image. A traffic signal is selected as being present when its confidence value is above a given threshold, unique for each traffic signal, and if it keeps a considerable distance from the confidence value of the other traffic signals. Only one traffic signal can win and the window for the occurrence of ties is relatively big. By default, if a tie occurs, the method results in the presence of an unknown traffic signal, see Fig.4.8.

By using template matching applied to the color image, it's intrinsically made a comparison on the context of shape and color of the object and how they co-exist on the image. For the various angles of view that can occur in a normal travel around the track, different template images are used for each type of traffic signal. This is done in order to reduce the variance and noise interference. The traffic lights have a black border around the traffic sign, so that border should be included in the template images in order to have a bigger template, and so achieve better and faster results. Small scale and rotational variances affecting the

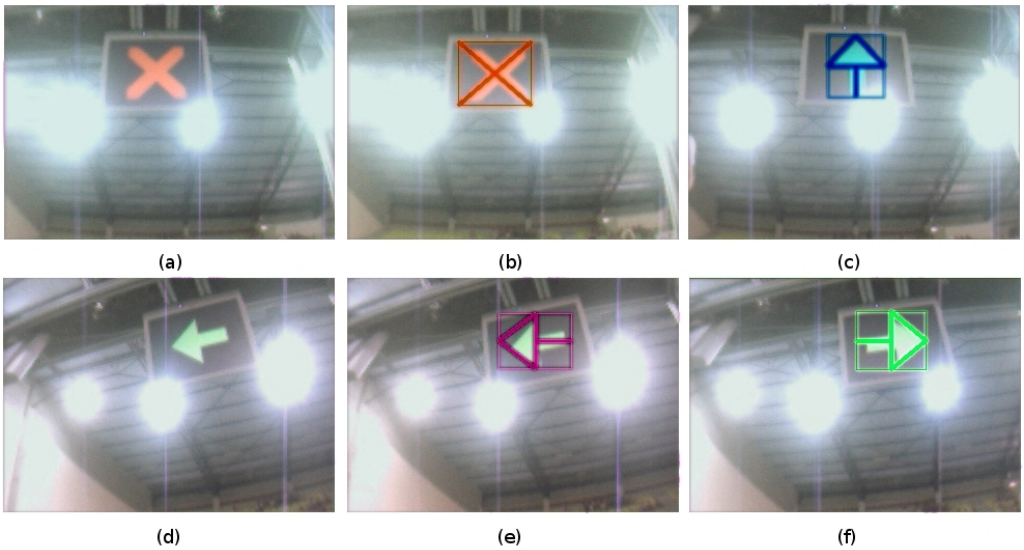


Figure 4.8: The traffic lights detection algorithm. For a visual debugging, a simple draw of the matched signal is displayed. The images (a) and (b) show the image before and after the detection. The same for (d) and (e), but here the car isn't aligned with the lane and therefore the signals appear rotated. Image (c) contains a detection of a up arrow, and (f) a right arrow.

system, didn't affected the final positive results as can be seen in Fig. 4.9. The algorithm proved to work in different scenarios and to have sufficient immunity to variance. The calibration process is quite fast since only a small image of the traffic signals needs to be gathered.

4.3 Road Maintenance Area Detection

In the current experimental platform, the detection of the road maintenance area represents one of the most challenging tasks. This area is a deviation from the original track, which is unknown beforehand, and is marked through the use of orange cones with white bands, connected with a plastic band with red and white stripes. When these elements are detected in the lane the robot must leave the original path and follow the new maintenance one. Due to the low position of the navigation camera, the detection of the plastic band is very difficult, since it is thin and appears in the horizon. Thus, the detection is based solely on recognition of the delimiting cones. The size of these cones on the image change drastically as the distance to them changes and their particular shape does not facilitate the recognition process. The problem is that if a square region of interest is considered for matching effects, the matching does not work, due to the changes in the image background behind the cones.

The implementation of a pattern matching method that considers only the shape of the cone would be rather difficult, since it must take into account the variability demanded by the detection in cause. The solution designed to deal with the problem is to use color segmentation followed by template matching. Using color segmentation, a search for orange elements is performed in the whole image. The segmentation should not be very restrict towards its

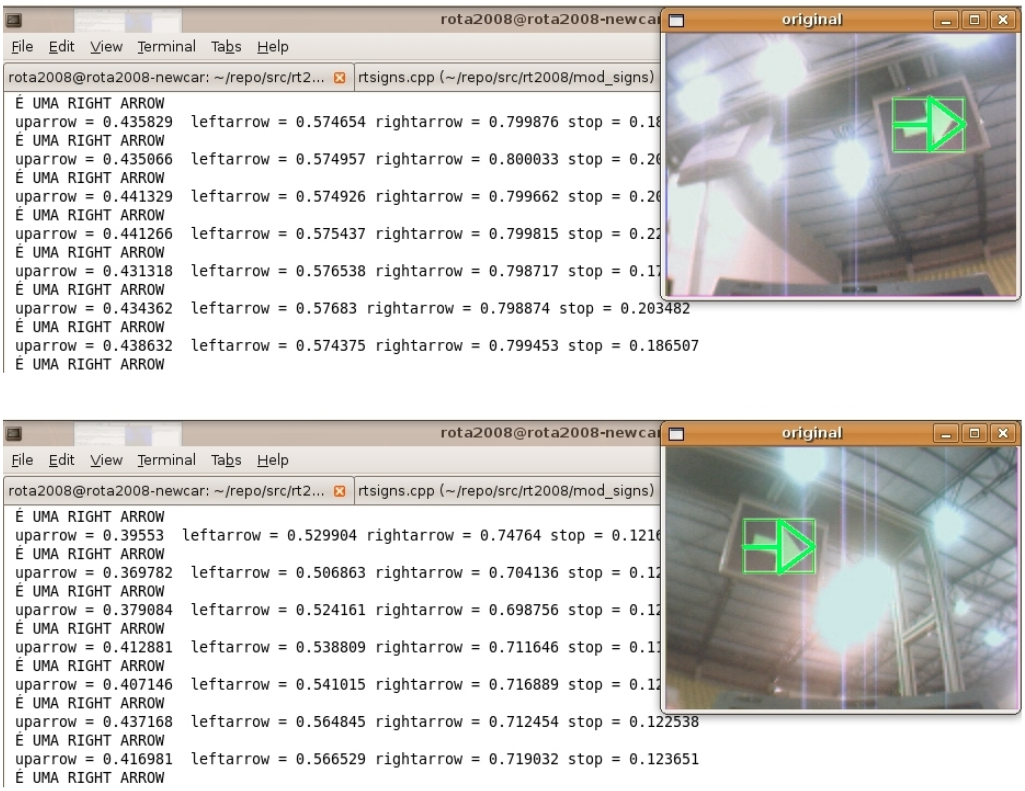


Figure 4.9: Two individual detections of a right arrow are displayed from two different angles. For each angle is shown the normalized matching scores for each one of the possible traffic light signs. That value is related to the confidence on the presence of that sign in the image.

results, by using a large window for the orange color. The segmentation is followed by template matching to retrieve the occurrences of the cones in the image.

The result of the color segmentation is a binary image where the area that isn't orange is set to "0". By using this approach, background interference is, in most of the cases, eliminated, so only the cones are considered for the matching, see Fig. 4.10.

It is still necessary to overcome the scaling issue derived from the range of possible cone sizes on the image. The use of scale invariant methods like SIFT [30], are not applicable for the current hardware and performance requirements. So, it was adopted a solution that consists in establishing image matching regions, associating a proper image template to each region.

The method works by searching the bigger selected regions of the image for bigger cones, since those are closer to the robot, and in smaller regions for smaller cones. The implementation is extensible to any number of matching regions, but as more regions are defined, the more computationally heavy the detection gets. This is important, since we have to guarantee that it can be executed within our performance requirements. So, with good practical results,

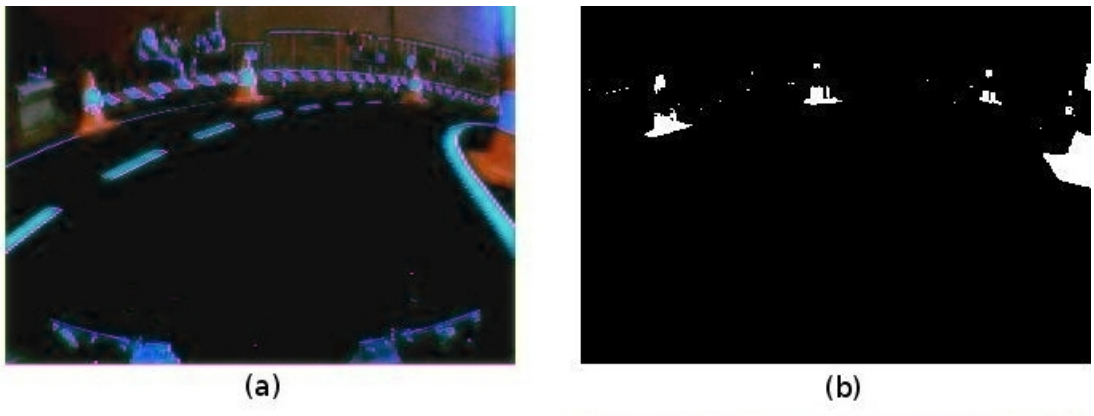


Figure 4.10: In the figure, it is used the orange color property of the cones in order to do background removal. The process used is color segmentation, where the original image (a) is segmented, resulting in image (b).

only two matching image regions are used.

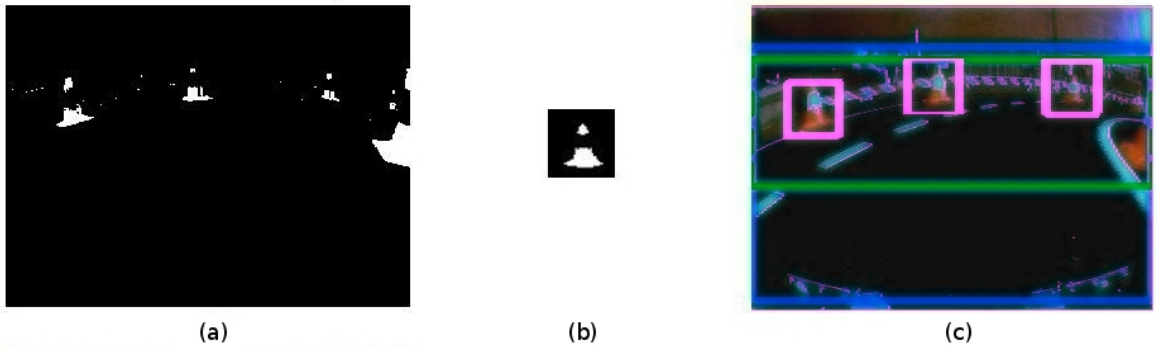


Figure 4.11: By using template matching on image (a) with the template (b), it is obtained a clear identification of the cones delimiting the road maintenance area (c). The matching region used is painted blue in this last image, the green only serves as example of how a bigger cones region matching can be. Bigger cones arise when they are more close to the robot.

As for the traffic light signals detection, the template matching is performed using the normalized correlation matching method, where higher scores translate in more confidence of a good matching. The detection of a cone is assumed successful if its matching score achieves a value above a given threshold. For each cone, there is a significant amount of pixels that achieves this score. So, in order to decrease the number of these multiple responses to a single cone, for the pixels that have the higher matching scores, the pixel's neighbourhood $w \times h$ is overwritten in the score vector. This, considering that $w \times h$ is the size of the template image being used in the current matching.

4.4 Obstacle Detection

The obstacle detection challenge, in this experimental platform, consists in identifying a green parallelepiped that occupies almost the full width of a lane, placed in a random position. The obstacle is always inside the road area, since it's supposed to represent a general lane obstacle. Thus, in order to search for the obstacle, it's considered only the road area, since all obstacles outside it are not of interest. Regarding this, it was developed a search procedure based on the detected lanes. The method consists on searching the lane space for high intensity transitions, using the Canny edge detector. It proved to be quite versatile, in the sense that any element placed in the lane would be considered and assumed as a lane object.

However, due to the low position of the road camera, the lane space that is seen is minimum and the view angle is such that sometimes an object could only be detected in a distance of 50-100 cm from the robot. Thus, efforts were conducted to combine the previous method with a detection based on the color of the object being searched. In order to detect the object, it's used color segmentation to search for image elements of the same color of the object. To filter the results and make the detection more robust, image morphology is used for noise-reduction, [45]. After the color segmentation, the opening morphological operator is applied to the image with a 3×3 standard kernel. This operator first applies erosions to the image followed by dilations. The resulting color blobs from the color segmentation process are first compressed and then expanded, so that smaller and dispersed blobs tend to disappear. The final result is the detection of the position and characteristics of the obstacle in the image and how it is related to the lane limits. The ROTA possess two directional sensors that result in poor identification of a obstacle that is not directly in front of the robot. After some tests, the performance of the computer vision algorithms easily overcome the limitations of the sensors measures.

Figure 4.12: In the figure, it is used color segmentation followed by the opening morphology operation. The result is a clear detection of the obstacle without noise. To finish the detection a scanning must be made, here showed in image (a).

4.5 Inverse Perspective Mapping

The Inverse Perspective Mapping is a technique widely used in the development of autonomous vehicles. There are many works that rely on this technique to achieve autonomous driving. The VisLab group (italian research group) has made accessible a very good description of their IPM system which is part of the GOLD vision system. In their technical report [9], they describe its use since the first ARGO vehicle project. Another good detailed description and implementation is found in [36].

4.5.1 Motivation

The IPM is used to map a relation between the image distances and real world distances, i.e., it allows to remove the perspective effect from incoming images remapping each pixel toward a different position in real distances with the assumption of a flat road in front of the vehicle. The result is a new two dimensional array of pixels (a remapped image) that represents a bird's eye view of the road region in front of the vehicle. Within the research included in the work supporting this thesis, it was found that the major groups that use this technique in autonomous vehicle developing, all exploit a knowledge about the specific acquisition parameters (camera orientation, camera position, optics of lens. etc...). That can be used in geometrical coordinate transforms, allowing to define general equations for the IPM use. However, the majority use high quality camera that have lens without distortions. The cameras used in ROTA have intrinsic distortion noise and a fish-eye mixed with radial lens property. This allows us to have a more wide view of the road in front of the car, but in the other hand, makes the determination of the camera parameters extremely difficult. In the referred state of the art platforms, the technique used consists in remapping the whole cameras image to remove the perspective effect. Then, the corrected images are used as the basis for the the low-level algorithms and the high-level lane detection. In our work, it's considered the inverse operation: first the image is submitted to the low-level and detections algorithms and then each interest point is remapped through an IPM look up table.

4.5.2 Method

Since the commonly used method for IPM, at a first glance, does not seem appliable to our context at least within the time restrictions of this work, due mainly, to the nonstandard nature of the ROTA's camera lens, another way was considered to achieve IPM. The method consists on determining the relation between image coordinates and real world distances, using real measures and interpolation. For the real measures, it was was developed the logic of measuring each component of the real coordinates in distinct phases. If the car is placed aside of a line parallel to the car's front axis, all the pixels where the line appears in the car's captured image are at the same horizontal distance. Putting the robot at different horizontal distances from the line, enough data is obtained, in order to interpolate the x component of the map. Data to interpolate the y component is obtained in a similar way, but now the car is placed facing the line.

Mapping Points

Any straight delimiting line can be used to collect the interpolation data. The detection of the line in the image is accomplished by the methods of global and adaptive search, also used for lane detection. So, one can choose all points that lie in the image line and then, by moving the robot several measures can be obtained while the search algorithm keeps tracking the line points. Each time the robot is moved, a new measure related to the robot mass center to the line is taken and introduced in a calibration software. This software, allows the use of an adaptive line search by defining any image region and direction with the use of the mouse. This allows to record measures for each component, load and save measure lists, scatterplot analysis (useful for easy analysis of an on-going calibration), record and erase of selective measures (including removing by sets), etc.

Bilinear Interpolation

When sufficient and correct measures are mapped for each x_r and y_r real world coordinate, it is used a bi-linear interpolation in order to map an estimation of the x_r, y_r components to the image pixels (x_i, y_i), see Fig. 4.13 and 4.14. Since only interpolations are used, due to the lack of trustworthy results obtained by means of extrapolation, it's required to define a value that symbolizes the absence of a value in the map look up table. The lack of map values occurs when, in the calibration process, the measures are taken based on points in the images borders, or when there is some unpredictable factor. Nevertheless, the lack of values in the mapping table defines the issue that consists in the fact that very small borders of the image don't have values, and is better to assume that there really isn't a value for that space, rather than assuming a wrong value result of some extrapolation. That situation has to be taken in account when using this method to achieve inverse perspective mapping.

Calibration

The measure process consists in obtaining a correspondence to the x_r and y_r in separate, but not by any specific order. Then, dislocating the robot by a determined distance to the line a new measure is taken. To ease the task, two A1 papers were glued together and printed to be used as a grid with a $5mm$ interval, see Fig. 4.15. The idea developed has its base that it is possible to use the own markings of the lane in the experimental platform to calibrate the IPM parameters, using the straight road section. The main advantage to this method is to be able to assign measures to around 80 image pixels (explanatory number) in one step, move the robot just a little, and do it again, while the software keeps tracking on the points to assign the measure.

Road Reconstruction in Real Distances

With the final interpolated pixel to real distances look up table, the IPM can now be used to map all the elements detected in the image to real world distances. Directly applied to the lane detection algorithm, the real environment distances in front of the car become fully available. By now, a direct approach to driving by road ahead can be accomplished robustly

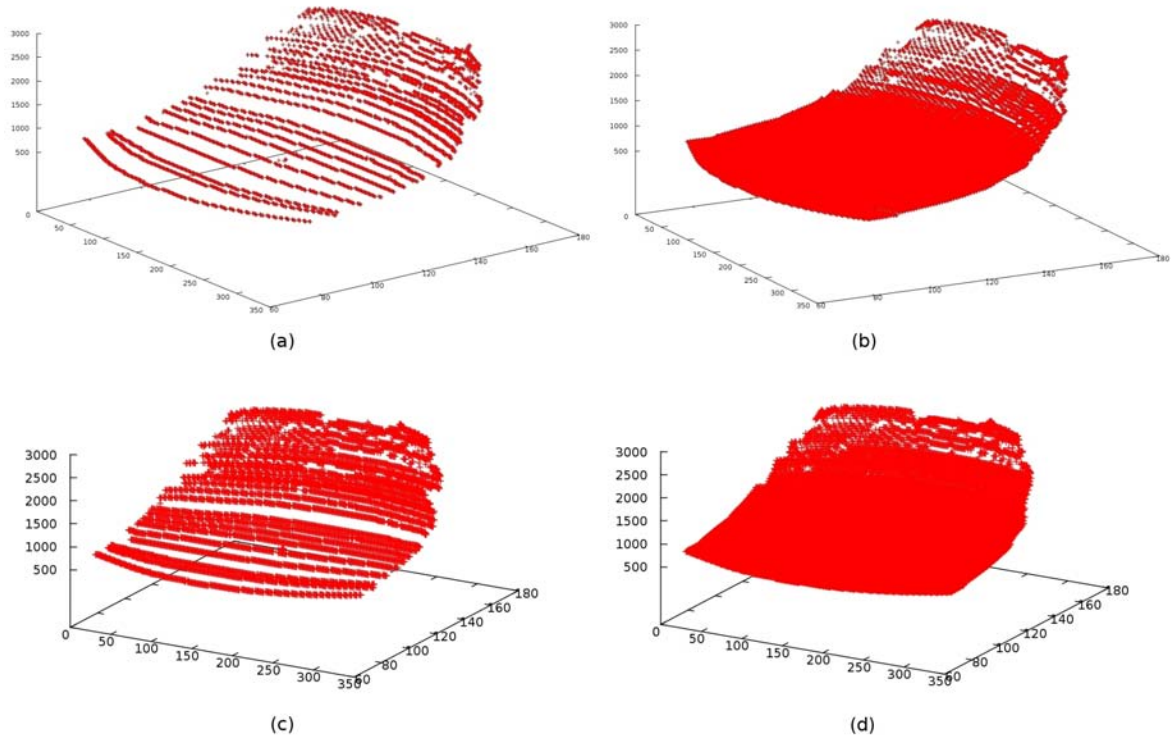


Figure 4.13: This figure contains several images results from the calibration plotting of y_r, as the independent value of x_i and y_i dependent ones. In the left images, (a) and (c) are the scatterplots for the measures taken during calibration. The assignment of the same measure to a line in the image can easily be seen here. The scatterplots represented in images (b) and (d) are the result of the bi-linear interpolation.

using the real world coordinate system. By keeping a direction to the middle of the lane in front of it, the robot can easily accomplish the task to maintain itself inside the desired lane by proportional control. The vision system combined with the use of the IPM, is enough to ensure that the vehicle has a high performance and never gets out of the track.

A new level of perception has been obtained, where real distances to all detection multi-purposes is kept independently, without the intervention of other processes ou behaviors. The front of the car environment is now fully known at any time, so one could choose to drive the robot by keeping a motion direction to a point in front of it that was inside the boundaries and didn't lay down on any lane object. The lane marking points can, in this case, be submitted to a regression analysis that can estimate and improve the real distances for better and wider results in the desired resolution. With regression analysis, a more complete representation of the lane markings is acquired. This is very useful for the case when one wants to compare the lane markings in the mapped, real world coordinate system.

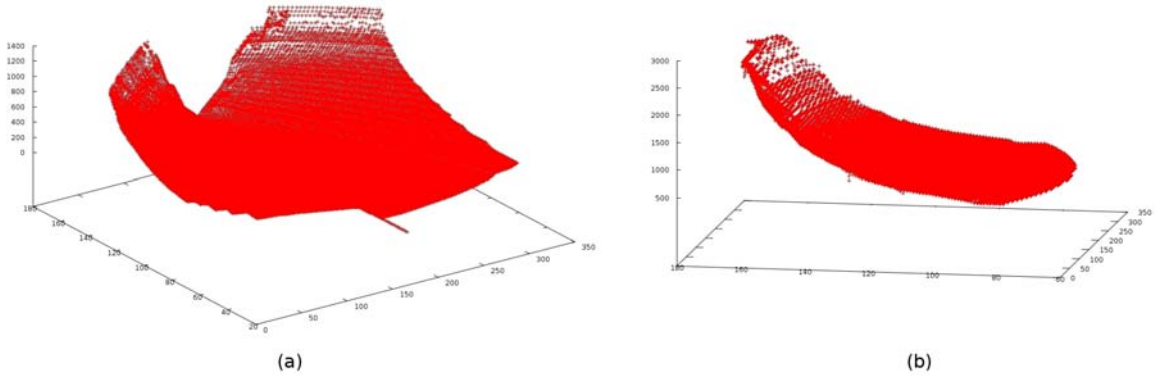


Figure 4.14: The resulting maps from the bi-linear interpolation are shown in scatterplots, where image (a) represents the x_r component and image (b), the y_r component. In the scatterplots xi and y_i, the image pixel coordinates, are the dependent variables in the X and Y axis. Note that for scatterploting x_r are only used the absolute values, for a better viewing, since by using the negative ones, a perception of the mapped values would be much harder.

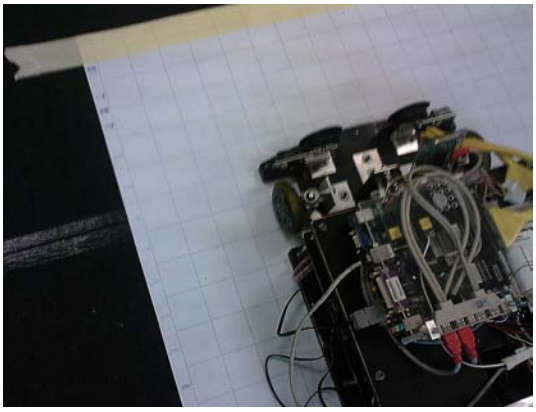


Figure 4.15: Robot car layed down on the grid used for calibration of the map for IPM. The grid's interval is $5mm$ for the light dark lines, and $10mm$ for the harder ones.

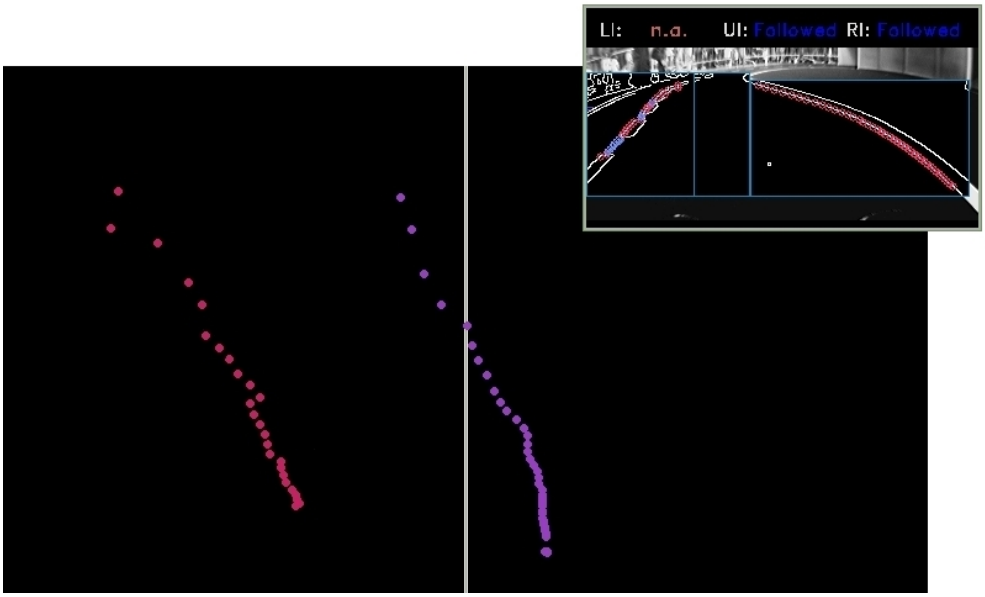


Figure 4.16: The continuous right and the discontinuous lane markings are detected as shown in the smaller upper right image. Then the structures pixel points are mapped through inverse perspective mapping and showed on the bigger black image, where the centered vertical line represents the car middle referential.

Chapter 5

Self Localization

Localization is considered a major issue when addressing mobile robotics, [39]. The goal of localization is to provide an autonomous robot with the ability to determine its position relatively to a known location in the environment, in order to navigate properly and be able to reach its goals. Many approaches already made towards autonomous driving, didn't really required any localization knowledge. In fact, within a constrained environment, an autonomous road vehicle do not necessarily needs to know where it is with respect to a geometric map, just like humans may not, since the problem can be approached by keeping within the road boundaries (the constrained environment), and any choices need only to be made at road junctions and where traffic signs are available.

Although the constrained nature of a road can allow a vehicle to achieve autonomous driving in a purely reactive fashion, it's feasible to provide it with positional data with respect to known locations in the environment to aid in navigation. Thus, allowing the vehicle to perform long term path planning. In order to do so, an autonomous vehicle needs to have the ability to answer the following questions:

- where am I?
- where do I want to go?
- how do I get there?

If an autonomous vehicle has the ability to answer these questions, it can properly calculate an optimal route (according to some criterion) and execute it within the known environment. With perfect sensing, this problem can be reduced to the minimization of the cost of a path between the start and end positions, while avoiding obstacles. However, when perfect sensing cannot realistically be achieved, the first question ("where am I?") cannot be properly answered, due mainly to some ambiguity in the perception of the environment (noisy sensors). For instance, in the case of an autonomous vehicle travelling on a highway, it can't perceive where it is by using only the visual information, because along the trip, road sections in different locations will look the same. Localization is therefore a critical issue, since if the robot does not know where it is, it cannot effectively plan for long term goals, or optimize recurrent trajectories.

This section focus on an approach that can provide ROTA the ability to answer the first question: "where am I?". The purpose is to allow the robot to know where it is relative to a specific location (for instance, the starting point, the crosswalk...), in order to make useful decisions in due time. If the robot car can successfully locate itself in the track, it can extrapolate a more smooth and optimal path. If road artifacts, such as obstacles or a road maintenance area, are encountered during a first lap, an improved route can be planned for the upcoming laps taking these obstacles into account. For instance, in the road maintenance area, the robot can go through it with increased confidence and speed.

The use of the visual perception system ensures that the available road markings are extracted, by using computer vision methods. The result of the perception layer is a tuple $< s, p >$ (shape, points) for each lane marking. This tuple contains the marking's shape, which can be a straight line or a curve, and contains the lane marking's detected points, mapped to a coordinate system centered on the robot's geometric center. The final purpose of the self localization is to estimate a position for the robot related to a global coordinate system. The map can be previously created offline, but also on-line while the robot explores its environment with any road artifact encountered added on the fly.

The method used for self localization uses a map generated by the visual perception system at the current position (the local track section's lane markings), to determine the pose of the robot relative to the road, which will be referred from now on as the local car pose. This local pose is compared to a previously generated map of the environment (a global map of the entire track), in order to estimate a global car pose that includes the robot's position relative to the global coordinate system. The estimation of the global pose also uses dead-reckoning, [39], which is a technique that integrates the motion history of the robot over time to determine the position changes from the starting location. The problem with dead reckoning is that since the new positions are obtained from the previous, the error is cumulative, leading to an erroneous position over time. The integration of dead-reckoning is achieved, by using an extended Kalman filter to probabilistically update the robot's position (global car pose).

The self localization method is based on a metric definition of an entire track's map. The can only measure the lateral offset and angle relative to a local track section using the visual perception system results. Since the track is divided in sections, first it is necessary to locate the robot car inside the current section and then, by relating the current section with the mapped track, locate the robot car in the track. The steps are:

- Use the position and orientation of the lane markings in order to obtain a localization for the robot car in the current track's section;
- Obtain the global car pose by relating the local car pose with the mapped track and the car dynamics.

The self localization of the robot can then be achieved at two levels of abstraction: the local car pose in the current road section, and a global car pose that relies on the first abstraction, track mapping and dead reckoning. In this chapter the methods developed for obtaining both abstraction levels are presented. The presentation begins with a description of the conceptual idea behind this approach, based on predicate logic, followed by refinements

and more detailed descriptions on the architecture models and system components. The last sections address the developments for obtaining the local and global car poses.

5.1 System Architecture

The overall architecture for ROTA's software can be seen as composed of three blocks or towers, as can be seen in Fig. 5.1. The "perception tower" is where the sensed environment is kept as a set of predicates that are processed through several layers of abstractions, aimed towards environment awareness through mapping and localization. This results in a description containing all environment elements and the robot car environment poses. The way these predicates are processed is by using inference rules and truth maintenance tables according to the models obtained on-line and off-line. The "model tower" makes the information available to both the "perception tower" as well as to the "action tower", so that all the information is centralized. The "action tower" deals with local and global planning by making use of the information state kept in the "model tower". This design is in some way similar to the architecture that Nils Nilsson described in his work entitled "Teleo-Reactive Programs and the Triple-Tower Architecture", [37], with the difference that he considered that the retrieving of the sensory apparatus is addressed by the "model tower". However, in this thesis context, the sensory retrieving is directly connected to the "perception tower" as depicted in Fig. 5.1. In the "model tower" is where the predicates abstractions obtained, as mapping and localization descriptions, are kept. The "perception tower" updates the information contained in the "model tower", making an effort to maintain it continuously faithful to the sensed environment. This information is to be used by the "action tower", to generate plans consisting of sequences of intermediate goals.

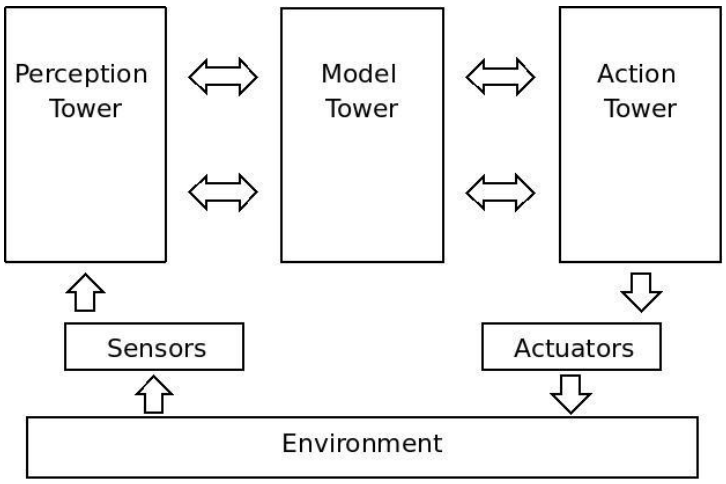


Figure 5.1: The proposed system conceptual model.

The main objectives of the proposed architecture are:

- Create standard definitions for track mapping and environment awareness in ROTA;
- Develop methods for estimating a global car position in the world and allow for track mapping. The track's map itself can be created off-line or on-line;
- Make use of dynamic perception, so that it uses knowledge about the robot actuations and therefore dead reckoning, to improve dynamically the environment awareness and confidence;
- Use planning to define way-points in a track in order to achieve long term intelligent planning.

5.1.1 Method

The perception tower retrieves the sensors information, processes and analyses it, in order to perceive where the car currently is on the road section (remember the road is always partially observed). After deciding a possible front road scenario by interpretation of the visible road, an extrapolation of each road delimiter is made in order to estimate the current robot car position related to the current road section. Then, a fusion is made in order to estimate a global pose with the prediction of the environment dynamics, using knowledge about the current motion of the car (dead reckoning), and the local car pose observations in the mapped track. The goal of perception is to update the model tower with a global position of the car in the world.

The action tower is where the modules responsible for global planning, motion planning and the control unit reside. This entity's goal is to define motion plans, that provide trajectories for the robot car. A control unit is used to execute the motion plans returned by a trajectory planner, it can also minimize the plans errors by using a proportional control system to drive the car to the specified way-points, making an effort to reduce the distance between the car position and the way-point position. The generation of the main way-points can be induced by a desire component. This component is considered also part of the action layer, and responsible in specifying the desires of our driving agent.

The functional model was later defined as an approach to the system overall implementation, where each entity in this model is typically a class in the OOP, Object-oriented programming paradigm, see Fig. 5.2. The arrows represent the retrieving of information or using of methods. This class model is to be used by the autonomous agent which life cycle consists on essentially three calls to the class model:

```
agent life cycle:
{
     WorldUpdater.update();
     Planner.update();
     Driver.drive();
}
```

The first call causes the perception to update the information about the environment contained in the world representation. The second call updates the motions plans, by retrieving information contained in the world representation. The planner can achieve long term planning, so the updates to the motion plans are not necessarily done at each cycle. To decide the trajectory, the world current state is considered at each call, so that there are changes in the long term plan if there are new constrains in the already planned path. The final call (*Driver.drive()*) applies the current motion plan to the car, in order to drive it to the next way-point. At each call, the current car state is checked for large deviations between the target way-point and the current car state, ensuring that the car does not get too far way from its goal.

The RoadPerceptor class, see Fig. 5.2 contains the vision methods for the handling of the front camera's images, where the road elements are found. The LightPerceptor class contains the methods needed to handle with the rear camera, used only for the traffic lights detection. The WorldUpdater class contains the methods for achieving self localization.

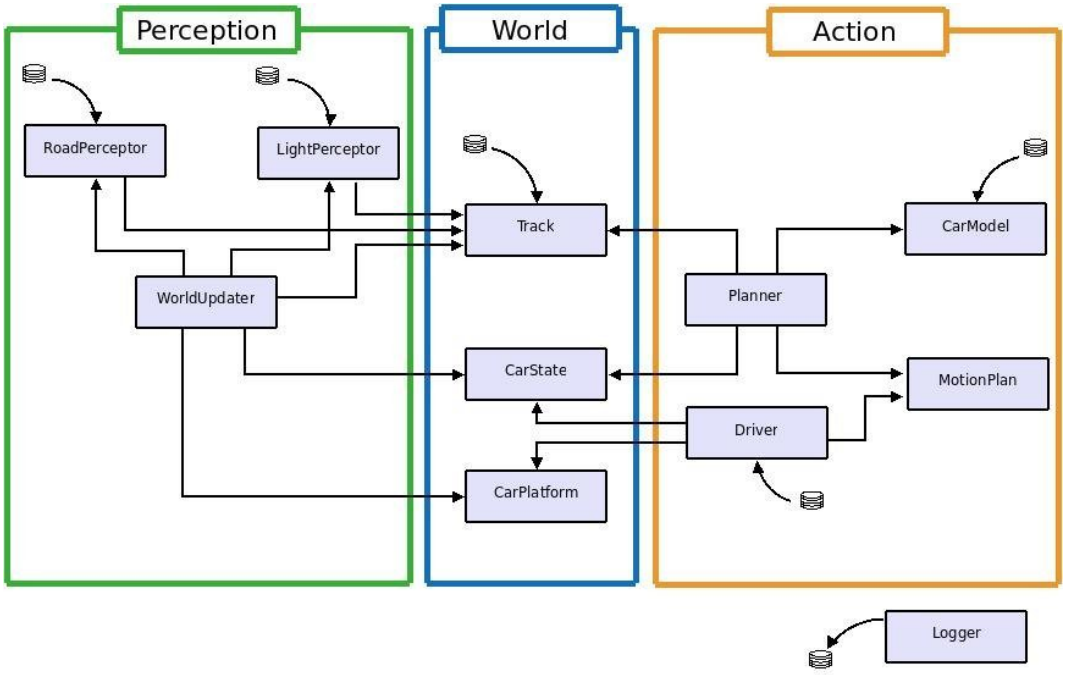


Figure 5.2: The proposed system class model.

5.1.2 World Representation

The purpose of this entity is to store information about the state of the environment. In it, it is defined the characteristics of the the car, its current state and information on the track, that can be mapped on-line or off-line. The role of this layer is to act as repository and bridge for the information flow in the architecture.

The CarPlatform class, see Fig. 5.2, contains several static characteristics of the car, i.e.,

that don't change along the agent's life cycle, such as the car dimensions (length, width, height), the wheelbase distance, maximum curving, maximum velocity. This class is regularly consulted by the perception and the action towers, and it's designed to provide re-usability. The idea is to maintain the usability of the software when changing the robot vehicle platform, so that in this case the only needed changes would be in the CarPlatform.

The Track class is suitable to organize and store information about the car's current track. The track is assumed to be made of individual sections that are connected together to form the complete track's representation. Sections have features like width, length or type: straight road, curved road, crosswalk, maintenance area, etc. The information can be updated on-line or off-line to contain objects or other information of interest.

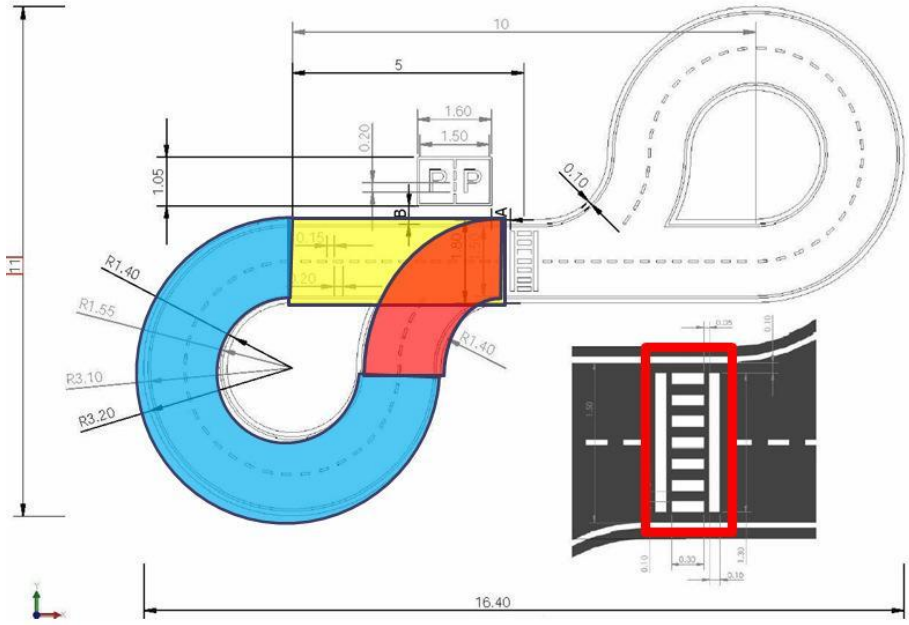


Figure 5.3: The track plant.

The 8-track and its sections definitions are represented in Fig. 5.3, recognizable by their color, the sections having real dimensions associated. The list of possible sections are:

- Straight Section
- Curved Section
- Crosswalk
- Road Maintenance Area
- Unknown Section

The unknown section type provides a way for the robot to map its own track, for self-mapping purposes. For the organization and management of tracks and its sections, a graph

approach was considered where each section is represented by a node and the arcs represent the possible directions between them, see Fig. 5.4. For the graph implementation the Boost Graph Library (BGL) [47] is used to deal with the graph component and C++ inheritance is used to instantiate the more specialized versions of the class Section. In the 8-track definition, as specified in the TrackModel, each node has its own features and are properly connected among them.

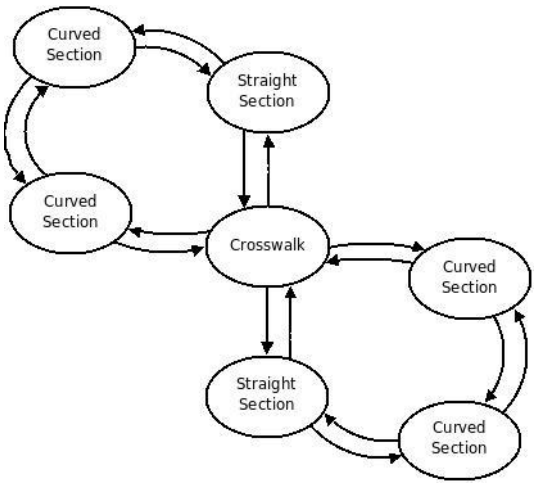


Figure 5.4: Track nodes for the 8-path used.

The CarState class is to be updated dynamically with the global coordinates of the car and motion information. The car direction in the track section is also kept to indicate if the car is driving in the considered positive direction or the reverse one. The attribute half-Laps included in this module is used to count the half-laps in the 8-track, being incremented each time the robot passes the crosswalk. The global pose of the car is given by a tuple $< x, y, \theta >$, where $< x, y >$ represents the cartesian coordinates of the reference point of the car (the center of its rear wheel) and θ is the car orientation (aka heading).

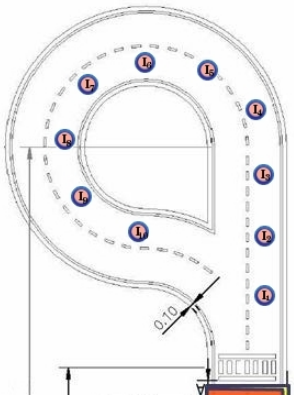


Figure 5.5: Example of how way-points can be previously defined.

A path generation, in the proposed system, consists on defining way-points in global co-

ordinates related to a mapped track. Associated to each way-point is information on how to perform the travel path between it and the next way-point. The Driver module is responsible for executing these way-point motion plans. If the car gets far away from the planned trajectory a new plan is requested. An example of how these way-points can be generated is in Fig. 5.5. Each way-point also contains the desired velocity at that point in the trajectory, so that the driver can later execute it properly. Between two way-points with different velocities is up to the Driver to decide how to make the transition.

5.2 Car Pose in a Local Road Section

By using the developed visual perception system, it's possible to estimate the lane's properties, including the type of section. The use of the IPM module, allows to map the road in front of the car in real distances, and by extrapolating it, a pose for the robot's geometric center, located somewhere near the middle of the rear wheel, can be estimated. This local pose, denoted by the pair $< x, \theta >$, is composed of a lateral offset (x) and an orientation angle (θ). The lateral offset represents the transversal deviation of the robot from the central delimiting line of the road. The orientation angle represents the angular orientation of the robot in relation to the same delimiting line. Note that in a curved section the central delimiting line is an arc of circumference and, thus, x is a radial deviation and θ an angle defined in relation to the tangent. Fig. **??** illustrate the meaning of x and θ in both a straight and a curved section. A longitudinal position of the robot can not normally be computed from the visual perception system, since, unless the robot is crossing a track's section border, the way it perceives the road at its front is always the same.

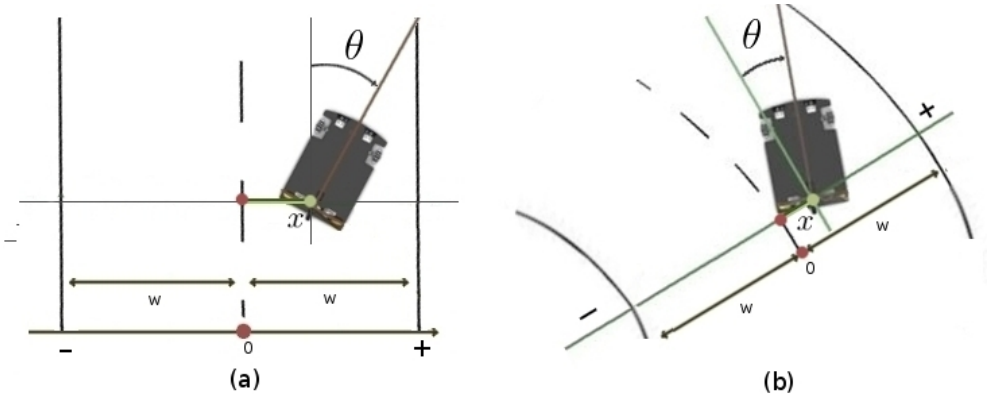


Figure 5.6: In image (a) is represented the road's coordinate system. The local pose of the car composed of the lateral offset and angle, $< x, \theta >$, are shown, as well as the width w for each lane. Image (b) contains the same local coordinate system in a curved section.

Straight Section

The method consists on the extrapolation of each lane marking in the front of the robot, so that the local car pose can be estimated, see Fig. 5.7. A linear regression is the most

appropriate method to approach the lane markings, where i represents each marking. The angle θ that the robot has for each marking, is as follows:

$$\theta = \arctan(m_i) \tag{5.1}$$

where m_i is the slope for each lane marking's regression. A line perpendicular to each lane marking passing in (0,0) is drawn and the euclidean distance between that point and the lane marking is measured. Let d_i be that distance which takes the negative form $d_i = -d_i$ if the lane marking is at the left of the robot. Being w_c the width of lane, the offset x is determined for each lane marking as follows:

$$x_i = \ell - d_i \tag{5.2}$$

$$\text{where, } \ell = \begin{cases} -w_c & \text{left marking} \\ 0 & \text{middle marking} \\ w_c & \text{right marking} \end{cases} \tag{5.3}$$

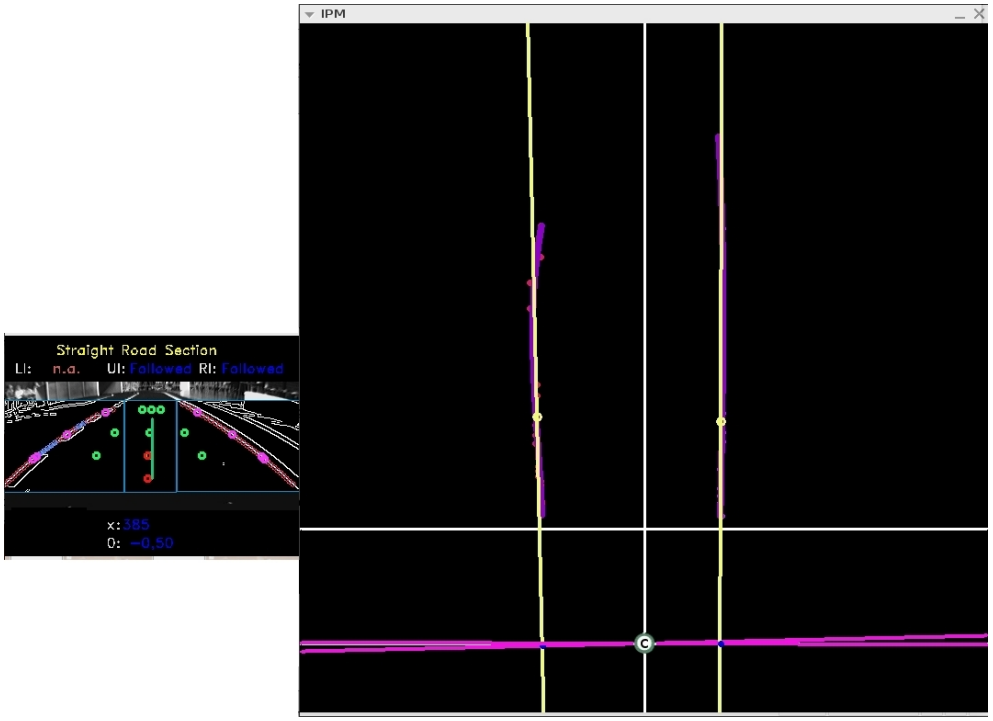


Figure 5.7: The straight section's markings are approached by linear regressions in order to estimate the car pose relative to its geometric center represented by (c). For the proper offset determination, perpendicular lines to the lane markings extrapolations that passes by (c), have to be considered. They are represented in the image by red/purple lines.

Curved Section

The solution for the case of a curved section is to approach the lane marking by an arc of a circumference. The iterative linear regression method didn't proved to be successful to

achieve this approach without an initial guess really close to the final solution. The final method used is based on geometry to determine the center and radius of the circle. First, each lane marking is divided into two equal or nearly equal parts, and the marking's points are split into two sets. For each set a linear regression is applied resulting in two line segments. Then, the perpendicular bisector of each segment is drawn, seen as white lines in Fig. 5.8. The intersection of the bisectors gives the center of the circumference. Formally, if $y = m_i x + b_i$, for $i = 1, 2$, are the equations of the perpendicular bisectors, then the center of the cincumference, $c = (c_x, c_y)$, is given by:

$$c_x = \frac{b_1 - b_2}{m_2 - m_1} \tag{5.4}$$

$$c_y = \frac{m_2 b_1 - m_1 b_2}{m_2 - m_1} \tag{5.5}$$

The radius r is obtained with an arithmetic mean of the Euclidean distances from the center of the circle to all the original points of the lane marking. Using the radius r and center pt, the distance d_i from $(0, 0)$ to each lane marking is as follows:

$$d_i = \sqrt{pt_x^2 + pt_y^2} - r \tag{5.6}$$

This distance d_i takes a negative value $d_i = -d_i$ if the lane marking is at the left of the robot. The angle θ is determined using $m_i = \frac{pt_y}{pt_x}$ in equation 5.1, and the lateral offset x using d_i in equation 5.2.

The current local car pose $< x, \theta >$ is the result of the weighted mean of the lane marking's lateral offset x_i and angle θ_i, by using their normalized confidences through the same process used for the determination of the section's type, in Chapter 4. The local car pose will be used for the estimation of the global car pose, as an instantaneous input measure.

5.3 Car Dynamics Model

In order to obtain the model that underlies the environment dynamics, one first need to establish the variables and referential in which the model will be based upon. At this point it is good to recall that the only measures available to determine the car dynamics are how much the car has travelled based on how much the rear wheel has turned, given by an odometry sensor, and the actuation angle in the steering wheels.

For the analysis, the three wheel robot car is considered as a two wheel one, which is possible since the robot mechanic conception was based on the Ackermann steering geometry, in which the linkages of the steering wheels are arranged so that it solves the problem for the different inside and outside angles of the wheels to correctly perform a turn. The difference between them is proportional to the degree of curvature that the car describes. Thus, the existence of a virtual wheel can be considered in the middle of both, and its angle represents the average value of the two steering wheels, see Fig. 5.9. In the figure, this virtual middle wheel can be seen, as for the distance between the robot front and rear wheels which is represented by L. When the car movement describes an arc, the center of the circle that contains that arc is represented by C, and the radius is R. In the current line of work it is considered that

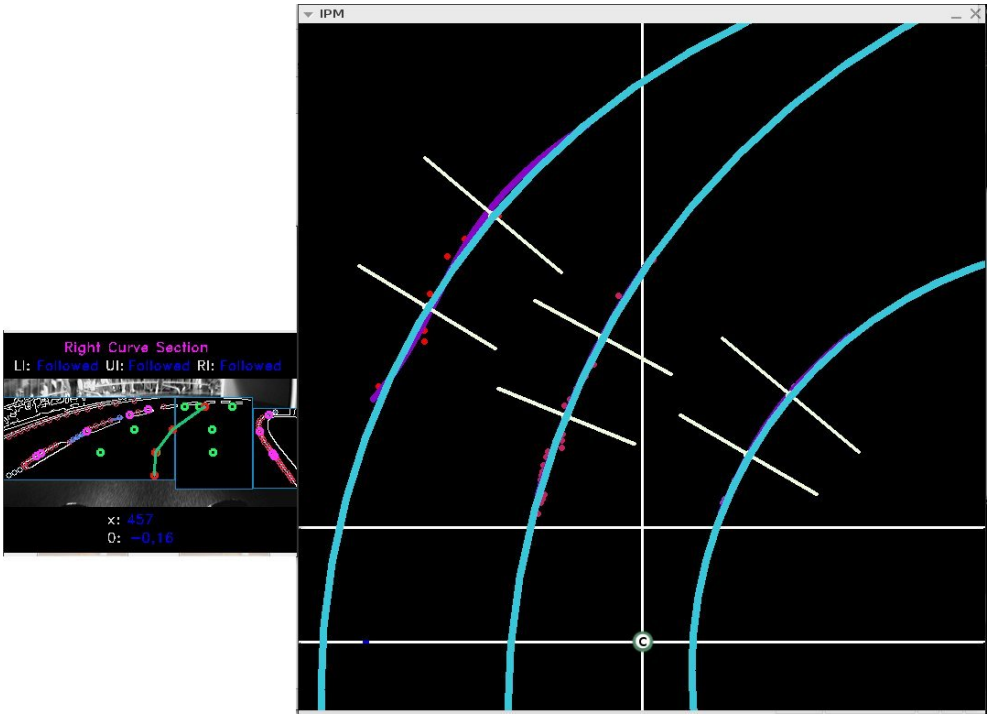


Figure 5.8: The lane markings are approach by arcs of circles. The center of the circles are given by the intersection of the two white lines that are perpendicular to each lane marking representation. The car pose is estimated relative to point (c).

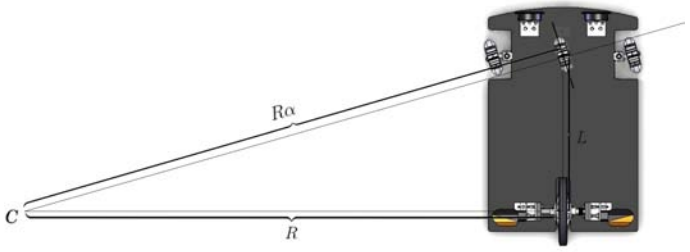


Figure 5.9: Robot curvature coordinate system. The virtual middle wheel considered can be seen between the two front ones.

the referential for R is positive from right to left. So, when the robot describes an arc to the left, as in the figure, the radius R is positive, for the case of an arc to the right, R is negative. The element $R\alpha$ is the distance from the virtual wheel to the center of the circle centered in C. The geometric center is situated in the middle of the rear wheel, and when describing a arc movement it is in this point where the origin of the coordinate system is positioned. This is done to best relate and establish a transformation model to describe the movement when performing a turn. Hence, an arc of a circle centered in C with radius R.

The use of R in the dynamic model brings the issue that when the car describes a path that is still an arc but tends to a rectilinear motion, the radius of the arc R of the motion

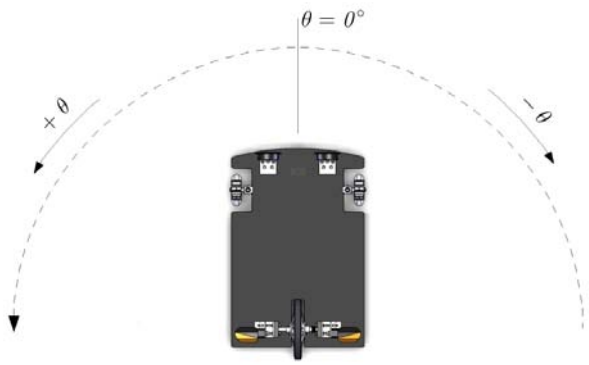


Figure 5.10: The car orientation given by θ referential.

described tends to infinite. In computer science it is best to deal with small number than large ones, and so the solution was to replace the radius R by the curvature $c = \frac{1}{R}$ which will be the final variable in the dynamic model. The global car position coordinates are always described by three variables, x, y, and angle θ.

The angle θ is required to describe the orientation of the robot in its environment, see Fig. 5.10. If the robot car is straight aligned vertically, as in the figure, the angle is 0. Note that this variable isn't related in any kind to the wheels steering or car curvature, its intention is to only describe the current orientation of the robot.

The dynamic model definition consists on a state space approach where the robot car has a initial state composed of a position (x_i, y_i, θ_i) and a motion plan description given by the curvature c and velocity v. The goal is to determine the final state given by position (x_p, y_p, θ_p) over time t. In which the state transitions are based on any impulse applied to the car through the control variables, which are the curvature c and velocity v.

Rectilinear Motion

A rectilinear motion of the robot is represented in Fig. 5.11, the initial and final positions are represented by (x_i, y_i, θ_i) and (x_p, y_p, θ_p), given the movement illustrated by the dashed arrow. In a rectilinear movement of the robot the angle θ does not change over time t. The analysis of the state change of the robot results in the following equations:

$$x_p = x_i - v\Delta t \sin\theta_i \tag{5.7}$$
$$y_p = y_i + v\Delta t \cos\theta_i \tag{5.8}$$
$$\theta_p = \theta_i \tag{5.9}$$

Curvilinear Motion

In a curvilinear motion where the robot is moving in a plane along a specified curved path and therefore has angular velocity, the front directional wheels are not aligned with the straight forward direction of the robot car and therefore $c \neq 0$ and $\theta_p \neq \theta_i$. A curvilinear

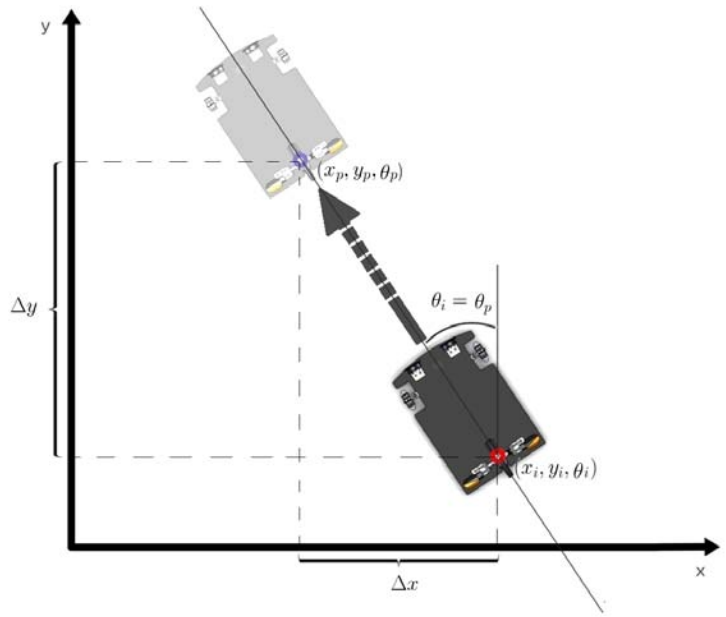


Figure 5.11: Car dynamics while describing a rectilinear movement.

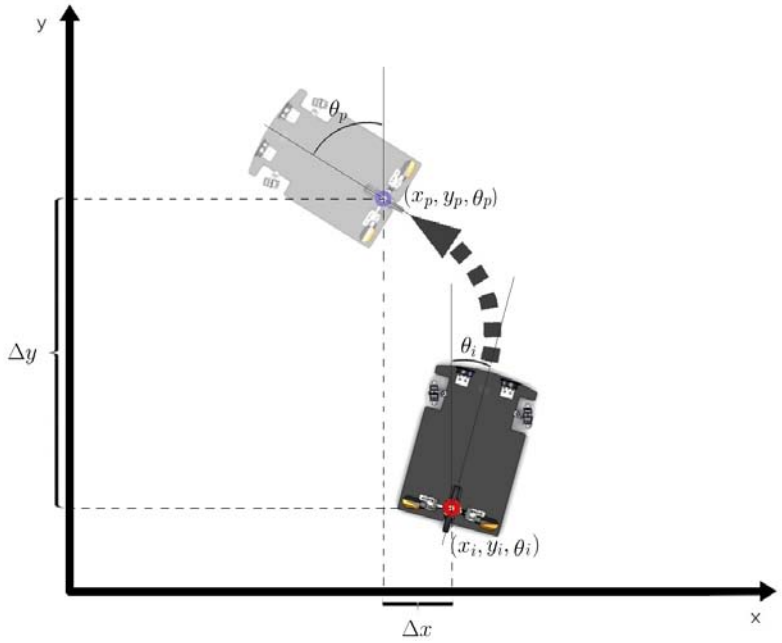


Figure 5.12: Car dynamics while describing a arc, that is a curvilinear movement.

motion is represented in Fig. 5.12 and the state changes in it are given by the following

equations:

$$\begin{aligned}\Delta x =& Rcos\left(\theta_i + \Delta\theta\right) - Rcos(\theta_i) \\ =& \frac{cos\left(\theta_i + v\Delta tc\right) - cos(\theta_i)}{c}\end{aligned} \qquad \begin{aligned}\Delta y =& Rsin\left(\theta_i + \Delta\theta\right) - Rsin(\theta_i) \\ =& \frac{sin\left(\theta_i + v\Delta tc\right) - sin(\theta_i)}{c}\end{aligned}$$

and so,

$$x_p = x_i + \frac{cos\left(\theta_i + v\Delta tc\right) - \cos(\theta_i)}{c} \tag{5.10}$$

$$y_p = y_i + \frac{sin\left(\theta_i + v\Delta tc\right) - \sin(\theta_i)}{c} \tag{5.11}$$

$$\theta_p = \theta_i + v\Delta tc \tag{5.12}$$

With the analysis of the robot car dynamic concluded, it is now possible to predict position changes along any motion, given initial positions, orientation, velocity and curvature. This useful approach provides a way to predict how the car is going to behave and therefore models its state transitions along time. This will be incorporated in the Kalman filter as a process transition model.

5.4 Car Pose in a Relative Coordinate System

Computer vision is the main embedded sensor in ROTA, still, the rear wheel is equipped with one odometry sensor that provides information about how much how far the robot has traveled, based on the amount that the wheel has turned. Unfortunately, the front steering wheels are not equipped with a sensor, so there is no way to measure their angle. With the independent perception layer developed, an intention to make a wealthier perception was brought by the idea that one can estimate how the car moves in the track by the study of the car dynamics and it's fusion with the perception layer. It is important to notice that such measurements are often inaccurate because of wheel slippage, surface imperfections, and small modeling errors.

5.4.1 Kalman for Self Localization

For global localization purposes, it's employed the use of an extended Kalman filter, since in this filter extension, the state transition and observation models need not be linear functions, but may instead be differentiable functions. In the thesis context, the car dynamics which will be in fact the transition model, as for the observation model described later in this section are not linear, and so a simple Kalman filter would not fit our purpose. By using an extended Kalman filter, we can make use of our car dynamics by predicting how the car should be according to the transition equations, and so the current measures of the world can be partially taken in account. The error associated with the deterioration of the confidence on the dynamic model along time is also deterministic, and incorporates the various process errors individually in an explicit way.

The aim of the "perception tower", in the scope of the proposed system architecture, is to update a state vector $\mathbf{x}_k$, which contains the global state of the car in the world. However, in the thesis context, the global states represented by the state variables $< Gx, Gy, G\theta, v, c >$ (global x,y,θ,velocity,curvature) can't be directly measured. The variables that one can estimate directly, are the local position in the road of the robot car, that is, $< lx,\ l\theta >$, and the odometry *odo* which is effectively measured, not estimated, by counting up how many the rear wheel has turned in a specific amount of time. The instant actuation on the front wheels as a curvature Δc can be also be taken in account, because this is effectively known, while the c real curvature is not. The same applies to the velocity v, which isn't measured, but the actuations on it Δv are known. Therefore, the aim of the whole process is to use the full potential of the Kalman filter with its ability to update the state vector with indirect measures, by means of observation models that relate them. This is achieve by incorporating the mapping component.

In order to apply the Kalman filter one first need to define the state vector $\mathbf{x}$, the control vector $\mathbf{u}$ and the observation vector $\mathbf{z}$. For the case in study the state vector was identified as:

$$\mathbf{x}_k = \begin{bmatrix} Gx \\ Gy \\ G\theta \\ v \\ c \end{bmatrix} \tag{5.13}$$

where $< Gx, Gy, G\theta >$, v,c represent, respectively the global pose, the velocity and the curvature of the car at time k. The control vector is defined as:

$$\mathbf{u}_k = \begin{bmatrix} \Delta v \\ \Delta c \end{bmatrix} \tag{5.14}$$

representing the increment in velocity and in curvature applied to the car at time k. Finally, the observation vector was defined as:

$$\mathbf{z}_k = \begin{bmatrix} odo \\ lx \\ l\theta \end{bmatrix} \tag{5.15}$$

where $< lx, l\theta >$ is the local car pose, at time k, and *odo* is the value returned by the odometric sensor also at time k.

Transition Model

The Kalman transition model is obtained by simply employing the knowledge about the robot car dynamics. The transition model is given by the following function f for the case of

a rectilinear motion:

$$f(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} Gx - v\Delta t \sin(G\theta) \\ Gy + v\Delta t \cos(G\theta) \\ \theta \\ v + \Delta v \\ c + \Delta c \end{bmatrix} \tag{5.16}$$

For the case of a curvilinear motion, the transition function f takes the following form:

$$f(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} Gx + \frac{\cos(\theta + v\Delta tc) - \cos(\theta)}{c} \\ Gy + \frac{\sin(\theta + v\Delta tc) - \sin(\theta)}{c} \\ \theta - v\Delta tc \\ v + \Delta v \\ c + \Delta c \end{bmatrix} \tag{5.17}$$

In the extended Kalman filter, the function f cannot be applied directly to the covariance matrix. Instead a state transition matrix F is computed which is defined to be the matrix of the partial derivatives (the Jacobian) of f. The intention in doing this is to linearize the non-linear model around the current state, so that it can be applied in to the covariance matrices.

$$F_{i,j} = \frac{\partial f_i}{\partial x_j} \tag{5.18}$$

The transition matrix F_k for the case of a rectilinear motion (where $c = 0$) is:

$$F_k = \begin{bmatrix} 1 & 0 & -v\Delta t \cos(G\theta) & -\Delta t \sin(G\theta) & 0 \\ 0 & 1 & -v\Delta t \sin(G\theta) & \Delta t \cos(G\theta) & 0 \\ 0 & 0 & 1 & \Delta tc & v\Delta c \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.19}$$

For the case where the robot car is describing a curvilinear motion (where $c \neq 0$), the Jacobian matrix F_k is:

$$F_k = \begin{bmatrix} 1 & 0 & \frac{\sin(G\theta) - \sin(G\theta + v\Delta tc)}{c} & -\Delta t \sin(G\theta + v\Delta tc) & \frac{-v\Delta t \sin(G\theta + v\Delta tc)}{c} - \frac{\cos(G\theta + v\Delta tc) - \cos(G\theta)}{c^2} \\ 0 & 1 & \frac{\cos(G\theta + v\Delta tc) - \cos(G\theta)}{c} & \Delta t \cos(G\theta + v\Delta tc) & \frac{v\Delta t \cos(G\theta + v\Delta tc)}{c} - \frac{\sin(G\theta + v\Delta tc) - \sin(G\theta)}{c^2} \\ 0 & 0 & 1 & \Delta tc & v\Delta c \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

With the transition matrix, the time update phase of the extended Kalman filter can be computed. This phase uses the car dynamics to project the estimates of the state vector and the error covariance matrix over time k. No measures are taken in account at this phase, only the previous state estimate x_{k-1} and the control input vector u_{k-1}:

$$\hat{x}_{k|pred} = f(x_{k-1}, u_{k-1})$$
$$P_{k|pred} = F_k P_{k-1} F_k^T + Q_k$$

$\hat{x}_{x|pred}$ - state estimate
$P_{x|pred}$ - error covariance estimate

Observation Model

It is considered, in the scope of the current world model, that the road is segmented into sections that can be either straight, or curved ones. These sections are characterized by a reference position that will be here addressed by xt, yt and θt. These variables describe where the reference of these sections are in the global coordinate system. In the case of a section described by $(xt, yt, \theta t, width, curvature)$, the aim is to measure the current state of the local car as it sees it $(odo, lx, l\theta)$, and then with this local positioning update the global car pose $(Gx, Gy, G\theta)$. This relation is not direct and it is achieved by using the observation model h, which is non-linear. Let $\mathbf{z}_k$ be the observation vector at time k, and $\mathbf{x}_k$ the state vector, the observation model h is such that:

$$\mathbf{z}_k = < odo, lx, l\theta > \tag{5.21}$$
$$\mathbf{x}_k = < Gx, Gy, G\theta, v, c > \tag{5.22}$$

with the observation noise v_k, the observation equation is given by:

$$\mathbf{z}_k = h(\mathbf{x}_k) + v_k \tag{5.23}$$

The knowledge of the exact values of the current section variables xt,yt, θt are intrinsic in the non-linear observation model h and so this model fully depends on the current accuracy of those values. Just like in the transition model where it is necessary to consider separated cases for a straight or a curvilinear movement, for the observation model there is also the need to consider in separate the case where the section is rectilinear and the case where it is curvilinear. So, in the case of a straight section, the representation of the relationship between the three sets of variables in the two coordinate systems are illustrated in Fig. 5.13.

Using the relation represented in Fig. 5.13, the observation model h can be determined. So, for the case of a straight section, that is, without curvature ($c = 0$), the observation model is as follows:

$$h(\mathbf{x}_k) = \begin{bmatrix} v\Delta t \\ G\theta - \theta t \\ (Gx - xt)\cos(\theta t) + (Gy - yt)\sin(\theta t) \end{bmatrix} \tag{5.24}$$

For the case of a curvilinear section, that is, where $c \neq 0$, the relation between the coordinate systems, and therefore the three set of variables is as represented in Fig. 5.14. By using these relations the observation model takes the following form:

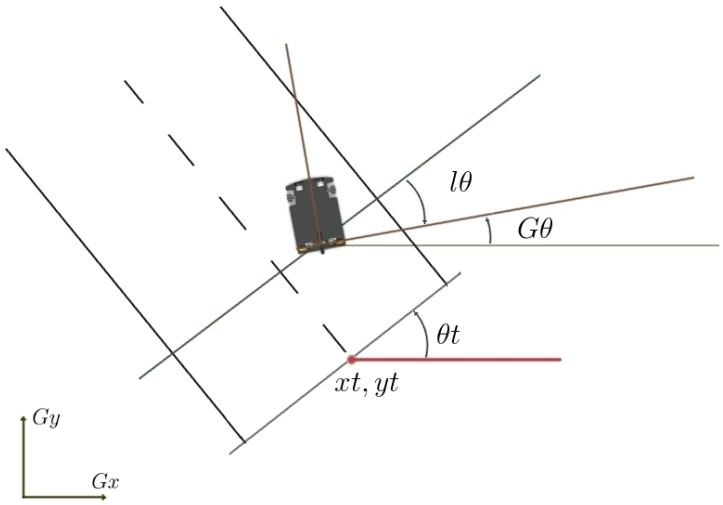


Figure 5.13: Observation model in a straight section. In the image, it is represented the relation between the local car variables that are estimated on-line, with the global car coordinates and with the section variables.

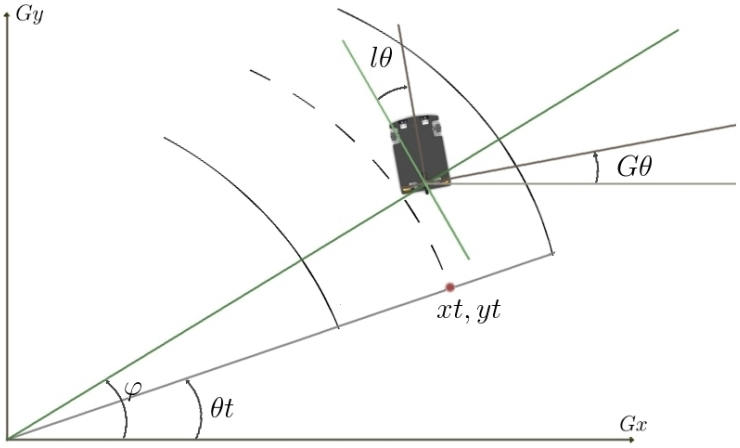


Figure 5.14: Observation model in a curvilinear section. In the image, it is represented the relation between the local car variables that are estimated on-line, with the global car coordinates and with the section variables.

$$h(\mathbf{x}_k) = \begin{bmatrix} v\Delta t \\ \sqrt{(Gx - xt)^2 + (Gy - yt)^2} \\ G\theta - \varphi - \frac{\pi}{2} \end{bmatrix} \tag{5.25}$$

where

$$\varphi = atan2(Gy - yt, Gx - xt) \tag{5.26}$$

At each time step k the function h is to be used to compute the predicted measurement from the predicted state $\hat{\mathbf{x}}_{k|pred}$. By comparing this predicted measure vector with the current measure vector $\mathbf{z}_k$ which is retrieved by the robot car at time-step k, the measure residual $\tilde{\mathbf{y}}_k$ is computed as follows:

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|pred}) \tag{5.27}$$

However, since h cannot be applied to the covariance directly, a matrix of partial derivatives (the Jacobian) has to be computed. The observation Jacobian matrix H is defined as:

$$H_{i,j} = \frac{\partial f_i}{\partial x_j} \tag{5.28}$$

The first-order partial derivatives the Jacobian H, for the case of a straight section ($c = 0$) are determined as follows:

$$H_k = \begin{bmatrix} 0 & 0 & 0 & \Delta t & 0 \\ \cos(\theta t) & \sin(\theta t) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.29}$$

For the case of a curved section ($c \neq 0$) the Jacobian H is as follows:

$$H_k = \begin{bmatrix} 0 & 0 & 0 & \Delta t & 0 \\ \frac{Gx-xt}{\sqrt{(Gx-xt)^2+(Gy-yt)^2}} & \frac{Gy-yt}{\sqrt{(Gx-xt)^2+(Gy-yt)^2}} & 0 & 0 & 0 \\ \frac{1}{1+(\frac{Gy-yt}{Gx-xt})^2} & \frac{1}{1+(\frac{Gy-yt}{Gx-xt})^2} & 1 & 0 & 0 \end{bmatrix} \tag{5.30}$$

With the observation function h and its Jacobian H determined one can now compute the measure update equations of the extended Kalman filter:

$$\begin{aligned} \tilde{y}_k &= z_k - h(\hat{x}_{k|pred}) & \hat{x}_k &= \hat{x}_{k|pred} + K_k \tilde{y}_k \\ S_k &= H_k P_{k|pred} H_k^T + R_k & P_k &= (I - K_k H_k) P_{k|pred} \\ K_k &= P_{k|pred} H_k^T S_k^{-1} \end{aligned}$$

$\tilde{y_k}$ - measure residual
S_k - measure residual covariance
z_k - measure vector
K_k - Kalman optimal gain
$\hat{x}_k$ - updated state estimate
P_k - updated estimate covariance

Self localization is accomplished by the following method: The robot car uses the visual perception to estimate a local car pose related to the section in front of him. Then, the previous global position and actuations in the world are used to obtain a predicted global position with the car dynamics. This predicted global position is related to the track map to know in which road section it currently is, so that the predicted global position can be transfortmed in a predicted measure using the observation model. The actual current measures are compared to the predicted ones to determine how the system should weight the current measures and the dead reckoning in the estimation of the current global state vector. The Kalman process is executed at each cycle, updating the global car pose estimates.

The concept is that by fusing the information provided by the vision system and dead reckoning, a more complete and accurate estimation of the relative robot position to a global coordinate system, centered in the track, can be established. So, in practical terms, it is expected that in the discreet time spaces where the vision system looses the tracking of the environment or has a large error associated, the use of the dynamic model to determine of how much the robot has traveled in the perceived environment relative to the position where he initially was, can result in the necessary compensation and ultimately help to correct the assumptions of the vision dependent perception layer.

Chapter 6

Conclusion and Future Work

This work focused on the development of a perception system. For that accomplishment, a deep study was held on Computer Vision and Robotics, specifically in the scope of Autonomous Driving. The development of a robot perception system, which inevitably only observes partially the surrounding environment, proved to be a very difficult task, with a high level of complexity associated. Many approaches have been developed within this scope but, still, none is truly faithful mainly due to the partial and noisy nature of the gathered data. Dealing with the observed world, and achieving a perfect mathematical model that can represent it, was never an easy task as result of the noisy nature of real world information. Furthermore, some variables of the observed world can't even be completely modelled and are assumed, for the sake of simplification, to be affected by a normal distribution noise. The sensors, specially low cost cameras, amplify these errors, by adding their own, and the resulting information requires heavy processing and handling. Recalling the proposed perception architecture in chapter 1 section, "Goals", the following conclusion can be established without doubts: the higher the desired abstraction of the representation of the environment, the higher the complexity and ambiguity associated.

The major part of the work described in the thesis, was implemented and tested, showing good results in affordable computational time. The self localization was only partially integrated, lacking the determination of the covariance error matrices for the observation and process models, so that the extended Kalman can work correctly. The author of this thesis believes that they can be determined by a trial and error approach. Tests were conducted in the robot car for the predict phase of the extended Kalman filter, i.e., the dead reckoning component. These tests showed that the first Kalman phase is correct and works with reasonable errors. For this testing, an interface software was developed, which was designed to be also used to test the observation model, and determine the associated error variances and covariances. If this is achieved, the proposed system representing an higher abstraction layer can be fully integrated.

The development of the Lane Detection System was one of the most time consuming tasks, but the results paid off since the system always worked in the experimental environment. To perform tests on the lane detection system, the car drove by classic control at its maximum speed, presenting a smooth trajectory, even if it goes along in the middle of the road, which is the worst-case scenario. It also proved to work when the car isn't aligned with the lane

($\pm 50^{\circ}$). Another advantage is for the cases when the lane markings get more or less observed, the system adapts the search and continues to follow their positions. The other detections shown to also improve the overall state of the art of ROTA robot car, with less time spent on calibration and an increased immunity to environment interferences.

In the scope of Perception and Intelligent Localization for Autonomous Driving, several studies can be performed, in order to improve the current system. Therefore, next are presented some topics considered worth of being approached for future work.

One future work could be the design and implementation of two other behavioral layers for the more reactive perception layers (remmember the proposed architecture in chapter 1, section: Goals). The contributions of each layer to the final behavior of the robot, can be determined by their current importance given the environment state, since each generates a list of possible solutions for the robot actuations.

The vision perception system can be improved by using multiple cues where different approaches to the same issue could be weighted for a final decision. For example, one can use color segmentation for the lane detection in parallel with the Canny algorithm, and then use error estimates to choose the best lane representation. Another option is to extend the developed vision system to accommodate not only the lane marking's variance error in the Kalman final estimation, but also the covariances between them. A future work could be also to apply the current software to a real scenario with a real car.

Bibliography

[1] VisLab - http://vislab.it/, homepage visited on May, 2009.

[2] Robotica2008 - http://robotica.ua.pt/robotica2008/ 8° Festival Nacional de Robótica visited on December, 2009.

[3] Git - http://git-scm.com/, Fast version control system visited on May, 2009.

[4] Doxygen - http://www.stack.nl/~dimitri/doxygen/, Source code documentation generator tool visited on May, 2009.

[5] Nicholas Apostoloff. *Vision based lane tracking using multiple cues and particle filtering.* PhD thesis, The Australian National University, 2005.

[6] Nicholas Apostoloff and Alexander Zelinsky. Robust vision based lane tracking using multiple cues and particle filtering. pages 558–563. IEEE, 2003. Intelligent Vehicles Symposium. Proceedings.

[7] José Luís Azevedo, Artur Pereira, Bernardo Cunha, and Luís Almeida. Rota: a robot for the autonomous driving competition. Technical report, Universidade de Aveiro, LSE-IEETA / DETI, 3810-193 Aveiro, Portugal, 2007.

[8] Massimo Bertozzi and Alberto Broggi. GOLD: a Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection. *IEEE Transations on Image Processing*, 7(1), 1998.

[9] Massimo Bertozzi, Alberto Broggi, and Alessandra Fascioli. Obstacle and lane detection on argo. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems'97*, pages 1010–1015, 1997.

[10] Massimo Bertozzi, Alessandra Fascioli, Corrade Guarino Lo Bianco, and Aurelio Piazzi. The argo autonomous vehicle's vision and control systems. In *International Journal of Intelligent Control and Systems*, pages 409–441, 1999. Dipartimento di Ingegneria dellInformazione Universit di Parma.

[11] Gary Bradski and Adrian Kaehler. *Learning OpenCV, Computer Vision with the OpenCV Library.* O'Reilly, 2008.

[12] Kenneth M. Brown and Jr. J. E. Denms. Derivative free analogues of the levenberg-marquardt and gauss algorithms for nonlinear least squares approximation. Technical report, IBM Philadelphia Scientific Center and University of Minnesota, 1972.

[13] R. Brunelli. *Template Matching Techniques in Computer Vision: Theory and Practice.* 2009.

[14] N.W. Campbell and B.T. Thomas. Lane boundary tracking for an autonomous road vehicle. 1992. Advanced Computer Research Centre University of Bristol.

[15] Octavia I. Camps, Tapas Kanungo, and Robert M. Haralick. Gray-scale structuring element decomposition. volume 5. IEEE Transactions on Image Processing, JANUARY 1996.

[16] John Canny. A computational approach to edge detection. pages 679 – 698. November 1986. IEEE Transactions on pattern analysis and machine intelligence, vol. PAMI-8, N.6.

[17] Takeo Kanade Chuck Thorpe, Martial Hebert and Steven Shafer. *Toward Autonomous Driving: The CMU Navlab. Part II: System and Architecture*, 6(1):44 – 52, August 1991.

[18] José Nuno da Silva Carvalho. Rota'2008: um robô para condução autónoma. Master's thesis, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, 2008.

[19] E. D. Dickmanns. Efficient computation of intensity profiles for real-time vision.

[20] E. D. Dickmanns. Dynamic computer vision for mobile robot control. 1988. 19th International Symposium and Exposition on Robots, Sydney, Australia.

[21] E. D. Dickmanns and A. Zapp. Autonomous high speed road vehicle guidance by computer vision. *Automatic Control World Congress*, pages 221–226, 1987. Selected Papers from the 10th Triennial World Congress of the International Federation of Automatic Control, Munich, Germany.

[22] Ernst D. Dickmanns. Dynamic vision-based intelligence. *AI Magazine*, 2004. AAAI.

[23] Ernst Dieter Dickmanns. *Dynamic Vision for Perception and Control of Motion.* Springer, 2007.

[24] Robert Fisher, Simon Perkins, Ashley Walker, and Erik Wolfar. *Image Processing Learning Resources.* 2004.

[25] John Fox. Nonlinear regression and nonlinear least squares. Appendix to An R and S-PLUS Companion to Applied Regression, January 2002.

[26] U. Franke, D. Gavrila, S. Görzig, F. Linder, F. Paetzold, and C. Wöhler. Autonomous driving approaches downtown. volume 13. 1999. nr.6 IEEE Intelligent Systems.

[27] Michael P. Georgeff and François Félix Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, August 1989.

[28] Pangyu Jeong and Sergiu Nedevschi. Efficient and robust classification method using combined feature vector for lane detection. volume 15. April 2005. nr.4 IEEE Transactions on Circuits and Systems for Video Technology.

[29] Junhyong Kim and Andrew Barron. Yale university department of statistics. http://www.stat.yale.edu/Courses/1997-98/101/, 1997. "Introduction to Statistics", Statistics Course 101-103, (66101,66103).

[30] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157, 1999.

[31] Bing Ma, Sridhar Lakshmanan, and Alfred Hero. Vision and navigation for the carnegie-mellon navlab. pages 362–373. IEEE, 1988. Pattern Analysis and Machine Intelligence.

[32] Bing Ma, Sridhar Lakshmanan, and Alfred Hero. Road and lane edge detection with multisensor fusion methods. pages 686–690. IEEE, 1999. Proceedings. International Conference on Image Processing.

[33] K. Madsen, H.B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems. Technical report, Informatics and Mathematical Modelling Technical University of Denmark, 2004.

[34] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 207(1167):187–217, 1980.

[35] Harvey Motulsky and Arthur Christopoulos. *Fitting Models to Biological Data using Linear and Nonlinear Regression, A practical guide to curve fitting.* GraphPad Software, 2003.

[36] Anuar Mikdad Muad, Aini Hussain, Salina Abdul Samad, Mohd. Marzuki Mustaffa, and Burhanuddin Yeop Majli. Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system. Technical report, Institute of Microengineering and Nanoelectronics, Universiti Kebangsaan Malaysia, 2004. IEEE.

[37] Nils Nilsson. Teleo-reactive programs and the triple-tower architecture. *Electronic Transactions on Artificial Intelligence*, 5:99–110, 2001.

[38] Mark S. Nixon and Alberto S. Aguado. *Feature Extraction and Image Processing.* Newnes, 2002.

[39] Clark F. Olson. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation*, 16:55–66, 2000.

[40] K. Parvati, B. S. Prakasa Rao, and M. Mariya Das. Image segmentation using gray-scale morphology and marker-controlled watershed transformation. *Discrete Dynamics in Nature and Society*, 2008.

[41] institution = Research Institute for Advanced Study, Baltimore Md year = 1960 R. E. Kalman, title = A New Approach to Linear Filtering and Prediction Problems. Technical report.

[42] Elizabeth Reis. *Estatística Descritiva (2 ed.).* Lisboa: Edies Slabo, 1994.

[43] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *American Statistician*, (42):59–66, 1988.

[44] Ch. Santhaiah, Dr. G. Anjan Babu, and Dr. M. Usha Rani. Gray-level morphological operations for image segmentation and tracking edges on medical applications. *International Journal of Computer Science and Network Security*, 9(7), 2009.

[45] J. Serra. Image analysis and mathematical morphology. Academic Press, 1982.

[46] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg, 2008.

[47] Jeremy Siek, Lie-Quan Lee, and Andrew Lumsdaine. (bgl) - the boost graph library - a c++ standardized generic interface for traversing graphs. http://www.boost.org/, 2001. Indiana University.

[48] Lei Tao, Fan Yang-yu, , and HUANG Lian-bing. Fast lane recognition based on morphological multi-structure element model. *Optoelectronics Letters*, 5(4), 2009.

[49] Charles Thorpe, Martial H. Hebert, Takeo Kanade, and Steven A. Shafe. Vision and navigation for the carnegie-mellon navlab. volume 10. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1988.

[50] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[51] M.A. Turk, D.G. Morgenthaler, K.D. Gremban, and M. Marra. Vits - a vision system for autonomous land vehicle navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):342–361, 1988.

[52] Eric W. Weisstein. http://mathworld.wolfram.com/LeastSquaresFitting.html A Wolfram Web Resource, 1999-2009. "Least Squares Fitting." From MathWorld.

[53] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[54] Joachim Wuttke. lmfit — a c/c++ routine for levenberg-marquardt minimization with wrapper for least-squares curve fitting. http://www.messen-und-deuten.de/lmfit/., 2004. based on work by B. S. Garbow, K. E. Hillstrom, J. J. More, and S. Moshier.